Advances in Intelligent Systems and Computing 522

Adam Grzech
Jerzy Świątek
Zofia Wilimowska
Leszek Borzemski *Editors*

Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II

Springer

Advances in Intelligent Systems and Computing

Volume 522

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within "Advances in Intelligent Systems and Computing" are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

Advisory Board

More information about this series at http://www.springer.com/series/11156

Adam Grzech · Jerzy Świątek
Zofia Wilimowska · Leszek Borzemski
Editors

Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II

Springer

Editors
Adam Grzech
Department of Computer Science
Faculty of Computer Science
and Management
Wrocław University of Technology
Wrocław
Poland

Zofia Wilimowska
Department of Management Systems
Faculty of Computer Science
and Management
Wrocław University of Technology
Wrocław
Poland

Jerzy Świątek
Department of Computer Science
Faculty of Computer Science
and Management
Wrocław University of Technology
Wrocław
Poland

Leszek Borzemski
Department of IT and Management
Faculty of Computer Science
and Management
Wrocław University of Technology
Wrocław
Poland

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-46585-2 ISBN 978-3-319-46586-9 (eBook)
DOI 10.1007/978-3-319-46586-9

Library of Congress Control Number: 2016951674

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This four-volume set of books contains the proceedings of the 37th International Conference on Information Systems Architecture and Technology, or ISAT 2016 for short, held during September 18–20, 2016 in Karpacz, Poland. The conference was organized by the Department of Management Systems and the Department of Computer Science, Wrocław University of Science and Technology, Poland.

The International Conference on Information Systems Architecture and Technology has been organized by the Wrocław University of Science and Technology since 1970s. The purpose of the ISAT conference is to discuss a state of the art of the information systems concepts and applications as well as the architectures and technologies supporting modern information systems. Contemporary organizations seem to be knowledge-based organizations and in connection with that information becomes the most important resource. Knowledge management is the process through which organizations generate value from their intellectual and knowledge-based assets. It is a management philosophy, which combines good practice in purposeful information management with a culture of organizational learning, in order to improve business performance. The computers are able to collect and select the information can make some statistics, and so on, but decisions have to make managers basing on their experience and taking into consideration computer support. An improvement of decision-making process is possible to be assured by analytical process supporting. Applying some analytical techniques, such as computer simulation, expert systems, genetic algorithms, can improve quality of managerial information.

One of the conference's aims is also to consider an impact of the knowledge, information, computing, and the communication managing technologies of the organization functionality scope as well as the enterprise information systems design, implementation and maintenance processes taking into the account various methodological, technological, and technical aspects. It is also devoted to the information systems concepts and applications supporting exchange of goods and services by using different business models and exploiting opportunities offered by Internet-based electronic business and commerce solutions.

ISAT is a forum for specialized disciplinary research, as well as on interdisciplinary studies that aims to present original contributions and to discuss different subjects of today's information systems planning, designing, development, and implementation. The event is addressed to the scientific community, people involved in variety of topics related to information, management, computer and communication systems, and to people involved in the development of business information systems and business computer applications.

This year, we received more than 110 papers form about 10 countries. Each paper was reviewed by at least two members of Program Committee or independent reviewers. Only 86 best papers were selected for oral presentation and publication in the 37th International Conference on Information Systems Architecture and Technology proceedings.

The book is divided into four volumes which splits papers into areas: Managing Complex Planning Environments, Systems Analysis and Modeling, Modeling of financial and Investment decisions, Risk Management, Project Management, Logistics and Market, Artificial Intelligence, Knowledge Based Management, Web Systems, Computer Networks and Distributed Computing, High Performance Computing, Cloud Computing, Multi-agent Systems, Internet of Things, Mobile Systems, Service Oriented Architecture Systems, Knowledge Discovery and Data Mining, Quality of Service, E-Business Systems.

We would like to thank the Program Committee and external reviewers, who were essential for reviewing the papers and ensuring a high standard of the ISAT 2016 Conference and its proceedings. We thank the authors, presenters, and participants of ISAT 2016, and without them the conference would not have taken place. Finally, we thank the organizing team for their efforts during this and previous years which have led to a successful conclusion of the conference.

Wrocław, Poland
September 2016

Adam Grzech
Jerzy Świątek
Zofia Wilimowska
Leszek Borzemski

ISAT 2016 Conference Organization

General Chair

Zofia Wilimowska, Poland

Program Co-Chairs

Zofia Wilimowska, Poland
Leszek Borzemski, Poland
Adam Grzech, Poland
Jerzy Świątek, Poland

Local Organizing Committee

Zofia Wilimowska, Chair
Leszek Borzemski, Vice-Chair
Adam Grzech, Vice-Chair
Jerzy Świątek, Vice-Chair
Arkadiusz Górski, Technical Editor, Conference Secretary
Anna Czarnecka, Technical Editor, Website Administrator
Agnieszka Parkitna, Conference Secretary
Anna Kamińska, Conference Secretary
Michał Kowalski, Technical Coordinator
Ziemowit Nowak, Website Support
Mariusz Fraś, Website Support

International Program Committee

Witold Abramowicz, Poland
Dhiya Al-Jumeily, UK
Iosif Androulidakis, Greece
Patricia Anthony, New Zealand
Zbigniew Banaszak, Poland
Elena Benderskaya, Russia
Leszek Borzemski, Poland
Janos Botzheim, Japan
Patrice Boursier, France
Wojciech Cellary, Poland
Haruna Chiroma, Malaysia
Edward Chlebus, Poland
Gloria Crisan, Romania
Marilia Curado, Portugal
Zhaohong Deng, China
Małgorzata Dolińska, Poland
Ewa Dudek-Dyduch, Poland
El-Sayed El-Alfy, Saudi Arabia
Naoki Fukuta, Japan
Piotr Gawkowski, Poland
Manuel Graña, Spain
Wiesław Grudzewski, Poland
Adam Grzech, Poland
Katsuhiro Honda, Japan
Marian Hopej, Poland
Zbigniew Huzar, Poland
Natthakan Iam-On, Thailand
Biju Issac, UK
Arun Iyengar, USA
Jürgen Jasperneite, Germany
Janusz Kacprzyk, Poland
Henryk Kaproń, Poland
Yannis L. Karnavas, Greece
Ryszard Knosala, Poland
Zdzisław Kowalczuk, Poland
Binod Kumar, India
Jan Kwiatkowski, Poland
Antonio Latorre, Spain
Gang Li, Australia
José M. Merigó Lindahl, Chile
Jose M. Luna, Spain
Emilio Luque, Spain
Sofian Maabout, France

Zygmunt Mazur, Poland
Pedro Medeiros, Portugal
Toshiro Minami, Japan
Marian Molasy, Poland
Zbigniew Nahorski, Poland
Kazumi Nakamatsu, Japan
Peter Nielsen, Denmark
Tadashi Nomoto, Japan
Cezary Orłowski, Poland
Sandeep Pachpande, India
Michele Pagano, Italy
George Papakostas, Greece
Zdzisław Papir, Poland
Marek Pawlak, Poland
Jan Platoš, Czech Republic
Tomasz Popławski, Poland
Edward Radosiński, Poland
Dolores I. Rexachs, Spain
José S. Reyes, Spain
Leszek Rutkowski, Poland
Gerald Schaefer, UK
Habib Shah, Malaysia
Jeng Shyang, Taiwan
Anna Sikora, Spain
Małgorzata Sterna, Poland
Janusz Stokłosa, Poland
Remo Suppi, Spain
Edward Szczerbicki, Australia
Jerzy Świątek, Poland
Eugeniusz Toczyłowski, Poland
Elpida Tzafestas, Greece
José R. Villar, Spain
Bay Vo, Vietnam
Hongzhi Wang, China
Leon S.I. Wang, Taiwan
Jan Werewka, Poland
Thomas Wielicki, USA
Zofia Wilimowska, Poland
Bernd Wolfinger, Germany
Józef Woźniak, Poland
Roman Wyrzykowski, Poland
Jaroslav Zendulka, Czech Republic
Bernard Ženko, Slovenia

ISAT 2016 Reviewers

Patricia Anthony, New Zealand
Zbigniew Antoni Banaszak, Poland
Elena Benderskaya, Russian Federation
Grzegorz Bocewicz, Poland
Leszek Borzemski, Poland
Jerzy Brzeziński, Poland
Wojciech Cellary, Poland
Krzysztof Cetnarowicz, Poland
Haruna Chiroma, Malaysia
Witold Chmielarz, Poland
Grzegorz Chodak, Poland
Robert Ryszard Chodorek, Poland
Kazimierz Choroś, Poland
Andrzej Chydziński, Poland
Gloria Cerasela Crisan, Romania
Mariusz Czekała, Poland
Pedro D. Medeiros, Portugal
Aldona Dereń, Poland
Grzegorz Dobrowolski, Poland
Ewa Dudek-Dyduch, Poland
Mariusz Fraś, Poland
Naoki Fukuta, Japan
Krzysztof Goczyła, Poland
Arkadiusz Górski, Poland
Manuel Grana, Spain
Jerzy Grobelny, Poland
Adam Grzech, Poland
Bogumila Hnatkowska, Poland
Maciej Hojda, Poland
Zbigniew Huzar, Poland
Natthakan Iam-On, Thailand
Przemysław Ignaciuk, Poland
Jerzy Józefczyk, Poland
Krzysztof Juszczyszyn, Poland
Adam Kasperski, Poland
Włodzimierz Kasprzak, Poland
Grzegorz Kołaczek, Poland
Zdzislaw Kowalczuk, Poland
Andrzej Kozik, Poland
Dorota Kuchta, Poland
Lumír Kulhánek, Czech Republic

Halina Kwaśnicka, Poland
Jan Kwiatkowski, Poland
Wojciech Lorkiewicz, Poland
Jose M. Luna, Spain
Lech Madeyski, Poland
Zbigniew Malara, Poland
Rafał Michalski, Poland
Zbigniew Nahorski, Poland
Jerzy Nawrocki, Poland
Peter Nielsen, Denmark
Tadashi Nomoto, Japan
Andrzej Nowak, Poland
Krzysztof Nowicki, Poland
Cezary Orłowski, Poland
Donat Orski, Poland
Piotr Pacyna, Poland
Michele Pagano, Italy
Agnieszka Parkitna, Poland
Marek Pawlak, Poland
Willy Picard, Poland
Jan Platoš, Czech Republic
Łukasz Popławski, Poland
Dolores Rexachs, Spain
Radosław Rudek, Poland
Jarogniew Rykowski, Poland
José Santos, Spain
Danuta Seretna-Sałamaj, Poland
Anna Sikora, Poland
Maciej Stasiak, Poland
Malgorzata Sterna, Poland
Janusz Stokłosa, Poland
Grażyna Suchacka, Poland
Remo Suppi, Spain
Joanna Szczepańska, Poland
Edward Szczerbicki, Poland
Paweł Świątek, Poland
Jerzy Świątek, Poland
Halina Tarasiuk, Poland
Kamila Urbańska, Poland
José R. Villar, Spain
Krzysztof Walczak, Poland
Zofia Wilimowska, Poland
Marek Wilimowski, Poland

Bernd Wolfinger, Germany
Jozef Wożniak, Poland
Roman Wyrzykowski, Poland
Jaroslav Zendulka, Czech Republic
Bernard Ženko, Slovenia
Maciej Zięba, Poland

Contents

Part I
Embedded Systems Design and Applications

Modification of Concurrent Design of Hardware and Software for Embedded Systems—A Synergistic Approach

Mieczysław Drabowski

Abstract The objective of this research is to present the concept of synergic approach to the problem of system synthesis, i.e. a combined solution to task scheduling and resource partition problems. The model and approach are new and original suggestions allowing design of hardware and also software controlling the performance of a computer system. This is an approach which we call a synergistic concurrent synthesis (s-co-synthesis). This paper shows the results of selected representative computational experiments into different instances of system this synthesis problems which prove the correctness of the synergic design concept and indicate methods solving these problems.

Keywords Co-design · Synergic · Scheduling · Allocation · Partition · Optimization · Ant colony optimization · Branch and bounded

1 Introduction. Coherent Co-synthesis of Computer Systems—Model and Method

The goal of high-level synthesis of computer systems is to find an optimum solution satisfying the requirements and constraints enforced by the given specification of the system. The following criteria of optimality are usually considered: costs of system implementation, its operating speed, power consumption and dependability. A specification describing a computer system may be provided as a set of interactive tasks (processes, functions). The partition of the functions between hardware and software is the basic problem of synthesis. Such partition is significant, because every computer system must be realized as result of hardware implementation for its certain tasks. In the synthesis methods so far, the software and hardware parts were developed separately and then connected in process the co-called co-synthesis,

M. Drabowski (✉)
Faculty Electrical and Computer Engineering, Cracow University of Technology,
24 Warszawska Street, Krakow, Poland
e-mail: drabowski@pk.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_1

which increased the costs and decreased the quality and reliability of the final product. The resources distribution is to specify, what hardware and software are in system and to allocate theirs to specific tasks, before designing execution details. The problems of tasks scheduling are one of the most significant issues occurring at the procedure synthesis of operating systems responsible for controlling the distribution of tasks and resources in computer systems. The objective of this research is to present the concept of synergic approach to the problem of system synthesis, i.e. a combined solution to task scheduling and resource partition problems. The model and approach are new and original proposals allowing design of hardware and software for performing operations of the computer system. This is approach, which we called a s-co-synthesis. This research shows the results selected of computational experiments for different instances of system s-co-synthesis problems proving the correctness of the synergic synthesis concept and shows the methods solving these problems. Due to the fact that synthesis problems and their optimizations are NP-complete we suggest meta-heuristic method: Ant Colony Optimization. S-co-synthesis of computer systems, as well as synergic design methodology their structures and scheduling procedures may have practical application in developing the tools for aided for rapid prototyping of such systems. Classical process of hardware and software synthesis [1] of embedded computer system consists of the following stages (Fig. 1):

- Specification of the designed system in terms of functional and behavioral—requirement and constraint analysis. System description in a high-level language, abstracting from the technical implementation of its modules.
- Resource partition—designing the best possible system structure.
- Task scheduling—designing the best possible system control.
- Allocation of the system functions (tasks) and resources (structure modules), adaptation of structure system and control modules and their integration.

The measure for an efficient implementation of a computer system is the degree of its modules utilization, minimized idle-time of its elements and maximized

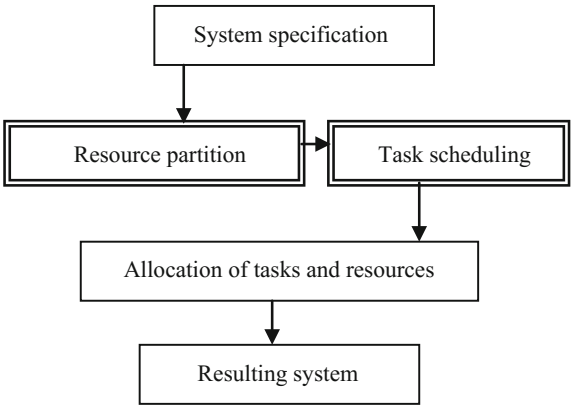


Fig. 1 General classical process of synthesis

parallel operation of its elements. A non-optimum system contains redundant modules or modules that are excessively efficient in comparison with the needs defined by the tasks which, consequently, increases the system cost. In high-level synthesis, the optimization of the designed system costs, speed and power consumption is usually an iterative process, requiring both changes in the architecture and task scheduling. Therefore, an optimum system should be created as a compromise between the projects: system control and its hardware organization.

1.1 General Model for the Problem of System Synthesis

System synthesis is a multi-criteria optimization problem. The starting point for constructing our approach to the issues of hardware and software synthesis is the deterministic theory of task scheduling [2]. The theory may serve as a methodological basis for multiprocessor and multisource system synthesis. Accordingly, the decomposition of general task scheduling model is suggested, adequate to the problems of computer system synthesis. Moreover, we wish to specify the model of task scheduling in a way suitable for finding optimum control methods (in terms of certain criteria)—as well as optimum assignment of tasks—in terms of other criteria —the division between universal and dedicated hardware components. Thus, we shall examine the system:

$$\sum = \{\mathbf{R}, \mathbf{T}, \mathbf{C}\} \tag{1}$$

where:

R—is the set of resources: hardware and software.
T—is the set of system tasks.
C—is the set of optimization criteria for the system's behavior and structure.

Resources. Recourse set consists of P processors = {P1, P2, …, Pm} and additional resources A = {A1, A2, …, Ap}.

Tasks. Set of tasks consists of n tasks which are to be processed on a set of m processors. A feasible schedule is optimal, if its length is minimal and it is implemented using minimum resource cost and minimal power consumption. Each task is defined by a set of parameters: resource requirements, execution time, ready time and deadline, an attribute—preemptable or nonpreemptable. The tasks set may contain defined precedence constraints represented by a digraph with nodes representing tasks, and directed edges representing precedence constraints. If there is at least one precedence constraint in a task set, we shall refer it to as a set of dependent tasks; otherwise we call it a set of independent tasks. The tasks form all the system functions, both outer practical and inner-operating, diagnostic and also transmission processes.

Criteria of optimality. As for the optimality criteria for the system to be designed, we shall assume its minimum cost, maximum operating speed and minimum power consumption [4]. We will apply multi-criteria optimization in sense of Pareto. The solution is optimized in sense of Pareto if it is not possible to find a better solution, regarding at least one criterion without deterioration in accordance to other criteria [5]. The solution dominates other ones if all its features are better. Pareto ranking of the solution is the number of solutions in a pool which do not dominate it. The process of synthesis will produce a certain number of non-dominated solutions. Although non-dominated solutions do not guarantee that they are an optimal Pareto set of solutions; nevertheless, in case of a set of sub-optimal solutions, they constitute one form of higher order optimal set in sense of Pareto and they give, by the way, access to the problem shape of Pareto optimal set of solutions. Let's assume that we want to optimize a solution of two contradictory requirements: the cost and power consumption Fig. 2. Synthesis of a system may also provide a system operating control, create an interface and provide methods and components for synchronization and communication between the tasks implemented by software and hardware. To sum up, the high-level synthesis of system, i.e. defining constraints and requirements of system, identifying its resources and operations, defining control should be implemented in **synergy** and be subject to multi-criteria optimization and verification during implementation.

1.2 Process of System Synergic Co-synthesis

Modeling the joint—synergic—search for the optimum task schedule and resource partition of the designed system into hardware and software parts is fully justified. We suggest the following schematic diagram of a coherent process of synthesis computer system—Fig. 3. Simultaneous consideration of these problems may be useful in implementing optimum solutions, e.g. the cheapest hardware structures. Synergic approach enables also performing all the assigned tasks with the minimum

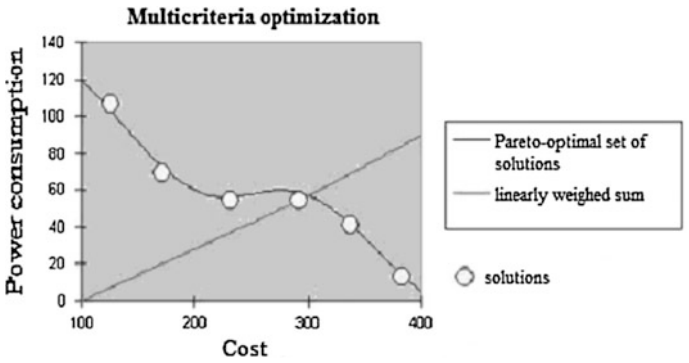


Fig. 2 Set of optimal solutions in sense of Pareto

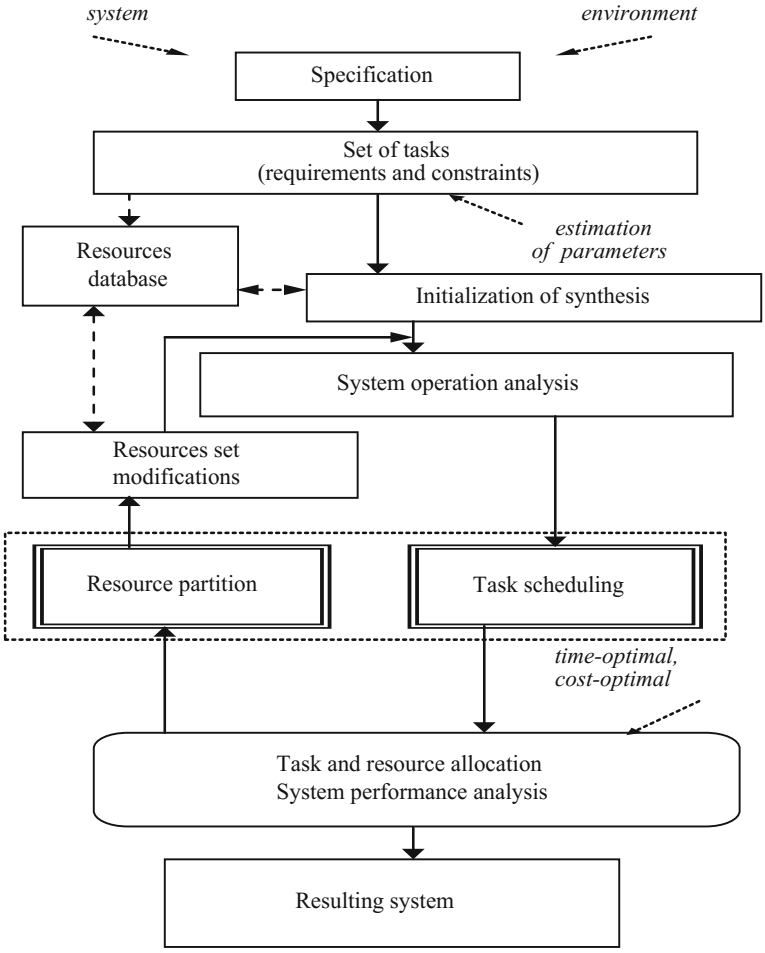


Fig. 3 Synergic process of synthesis of computer system: par-synthesis

schedule length. With such approach, the optimum task distribution is possible on the universal and specialized (dedicated) hardware and defining resources with maximum efficiency.

Assuming the initial values of resource set and task scheduling—initial resource set and task schedule should be admissible (from the pool of resources and schedules and from the historic data base remembered due to the synthesis of systems similar to them in the past); i.e. they should meet all the requirements in a non-optimum way. Task scheduling, resource partitioning and task and recourse allocation—tasks of specification, resources currently selected and assigned to certain tasks. Evaluating the operating speed, power consumption and system cost, multi-criteria optimization. The evaluation should be followed by a modification of the resource set, as a result of a new system partitioning into hardware and software

parts or achievement of a satisfying result. Iterative calculations are executed till satisfactory design results are obtained—i.e. optimal (or sub-optimal) system structure and schedule. The designed system should be fast, cheap, worth low power consumption and dependable.

1.3 Adaptation of Ant Colony Optimization Algorithm to Solve the Problems of Par-Synthesis

The Ant Colony Optimization (ACO) algorithm [6] is a heuristics using the idea of agents (here: ants) imitating their real behavior. Basing on specific information (distance, amount of pheromone on the paths, etc.) ants evaluate the quality of paths and choose between them with some random probability (the better path quality, the higher probability it represents). Having walked the whole path from the source to destination, ants learn from each other by leaving a layer of pheromone on the path. Its amount depends on the quality of solution chosen by agent: the better solution, the bigger amount of pheromone is being left. The pheromone is then "vapouring" to enable the change of path chosen by ants and let them ignore the worse (more distant from targets) paths, which they were walking earlier [7].

To adapt the ACO algorithm to synthesis problems, the following parameters have been defined:

- Number of agents (ants) in the colony,
- Vapouring factor of pheromone (from the range (0; 1)).

The process of choosing these parameters is important and should consider that:

- For too big number of agents, the individual cycle of algorithm can last quite long, and the values saved in the table ("levels of pheromone") as a result of addition will determine relatively weak solutions.
- On the other hand, when the number of agents is too small, most of paths will not be covered and as a result, the best solution can long be uncovered.

The situation is similar for the vapouring factor:

- Too small value will cause that ants will quickly "forget" good solutions and as a result it can quickly come to so called *stagnation* (the algorithm will stop at one solution, which doesn't have to be the best one).
- Too big value of this factor will make ants don't stop analyze "weak" solutions; furthermore, the new solutions may not be pushed, if time, which has passed since the last solution found will be long enough (it is the values of pheromone saved in the table will be too big).

The algorithm defines two more parameters, which let you balance between:

- α—the amount of pheromone on the path, and
- β—"quality" of the next step.

These parameters are chosen for specific task. This way, for parameters:

- $\alpha > \beta$ there is bigger influence on the choice of path, which is more often exploited,
- $\alpha < \beta$ there is bigger influence on the choice of path, which offers better solution,
- $\alpha = \beta$ there is balanced dependency between quality of the path and degree of its exploitation,
- $\alpha = 0$ there is a heuristics based only on the quality of passage between consecutive points (ignorance of the level of pheromone on the path),
- $\beta = 0$ there is a heuristics based only on the amount of pheromone (it is the factor of path attendance),
- $\alpha = \beta = 0$ we'll get the algorithm making division evenly and independently of
- the amount of pheromone or the quality of solution.

Having given the set of neighborhood N of the given point i, amount of pheromone on the path h and the quality of passage from point i to point j as an element of the table η you can present the probability of passage from point i to j as. Formula evaluation of the quality of the next step in the ACO algorithm– Formula 2.

$$P_{\ddot{y}}^{k} = \begin{cases} \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k}\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}} & when j \in N_i^k \\ 0 & else \end{cases} \tag{2}$$

In the approach presented here, the ACO algorithm uses agents to find three pieces of information:

- the best/the most beneficial division of tasks between processors,
- the best sequence of tasks,
- searching for the best possible solution for the given distribution.

Agents (ants) are searching for the solutions which are the collection resulting from the first two targets (they give the unique solution as a result). After scheduling, agents fill in two tables: two-dimensional table representing allocation of task to the given processor and one-dimensional table representing the sequence of running the tasks. The computational complexity of single agent process is polynomial and depends on the number of tasks, resources and times of tasks beginning. After initiating the tables (of allocation and sequence) for each agent, the algorithm starts the above cycle, after which the evaluation of solutions takes place. Having completed the particular number of cycles, the parameters are being updated and algorithm continues working:

1. initiation of tables of tasks running sequence and allocation of tasks to resources,
2. completing the cycle of analysis for each agent, evaluation of the best solution found in current cycle,

3. for each agent—basing on the best solution—updating the tables of tasks running sequence and allocation of tasks to resources,
4. is it the last cycle? (if not go to 2),
5. optimization/customization of system parameters.

2 Customization of the Branch and Bound Greedy Algorithm to Synthesis Problems Solving

Branch and Bound (B&B) algorithm [8] is a greedy algorithm browsing the set of solutions and "pruning" these branches, which give worse solutions than the best solution already found. This kind of approach often significantly reduces the number of solutions, which must be considered. However in the worst case scenario, "pruning" the branches is impossible and as a result, the B&B algorithm analyzes the complete search-tree. Both forms (DFS and BFS) of B&B algorithm were used for synthesis. It let us comprehend the problem of analysis of three different kinds of optimization (cost, power, time) without discrediting any of the problems. B&B algorithm investigates the problem by: choice of the task, definition of initial time to which you can schedule the task, choice of processor on which the task will be allocated. Because allocating the chosen task in the first available time unit or on the first available processor is not always the best idea, all available time units and processors are being considered. As a result, calculative complexity of algorithm changes exponentially when new tasks are added or polynomial after addition of new processors. Although B&B algorithm operation process is relatively simple, the number of solutions, which must be examined, is huge.

3 Calculative Experiments

Because one algorithm creates unlimited cycle and the other one takes a very long time to finish in many cases, the results given in the tables present state of the system after not more than three minutes of analysis. Depending on the solution criterion, there were used both forms of B&B—DFS and BFS—for the algorithm to be able to find a good solution in time.

Each solution given by Ant Colony algorithm will be graded on the basis of solutions found by Branch and Bound (B&B) algorithm. Formula for the quality of obtained solution is following—Formula 3. The final grade is influenced only by these parameters, which were being optimized by algorithms: cost, power and time of scheduling [4, 5]. The total quality of proposed system includes all three parameters (scheduling time, cost and power consumed by the system):

- the quality higher than 100 % means that ACO algorithm has found better solution than B&B,
- the quality equal 100 % means that both algorithms have found equally good solutions,
- the quality less than 100 % means that B&B algorithm has found better solution.

$$quality = 100\% \frac{1}{criterions} \sum_{criterion=1}^{criterions} \frac{result_{B\&B}}{result_{ACO}} \tag{3}$$

where:

criterions—number of criteria, $results_{B\&B}$—result obtained in the algorithm B&B etc.

The correctness of scheduling proposed by ACO and B&B algorithms was verified on the basis of the following examples. The algorithms were given the following resources: processors (general Processors and dedicated (Application-Specific Integrated Circuit)—specifications in Table 1, operating memory, mass storage.

3.1 Optimization of Scheduling Length and System Cost

Optimizing two aspects of system is much more difficult for the algorithms than minimizing a single parameter.

Because Pareto optimization in this case limits significantly our possibilities of finding the best system—as a criterion of the choice we can take the quality of obtained solution. Time, which has passed until solution was found and the parameters of the target system are presented in the Table 2. In the multi-objective optimization it is clear that ACO algorithm exceeds the greedy algorithm B&B in relation to the quality of solutions: solutions proposed by ACO algorithm are better

Table 1 Specifications processors

Id	Computational power	Consumption power (active)	Consumption power (non active)
Processor 1	1	100	10
Processor 2	2	120	12
Processor 3	4	120	15
Processor 4	8	200	20
ASIC 1	1	80	8
ASIC 2	2	110	11
ASIC 3	4	150	15
ASIC 4	8	180	18

Table 2 Parameters of the target system (optimization 3.1)

Number of tasks	Ant colony				Branch and bound				Quality
	Time	Length	Cost	Power	Time	Length	Cost	Power	%
5	11.1	12	9.8	912	0.5	6.0	30.5	4173	116.6
10	12.8	15	11.7	1338	0.2	7.9	30.51	2001	126.4
15	14.7	11	13.7	2953	0.5	4.5	29.11	3094	131.3
20	59.0	10.0	15.0	4007	0.4	6.0	31.09	4173	118.3
25	60.1	9.8	20.5	5054	0.1	7.9	32.11	5282	124.5
30	12.5	11.3	19.0	6057	0.9	10.1	29.78	6396	125,5
35	16.2	12.5	19.2	7004	0.8	11.3	30.51	7448	125.4
40	15.5	15.0	19.0	8010	0,6	13.5	30.7	8609	125.3
45	42.4	16.6	18.7	9011	0.5	15.1	29.9	9614	125.7
50	26.5	18.1	21.3	10,024	0.4	16.2	31.4	10,693	126.5
55	34.5	20.1	22.1	11,009	0.4	18.1	30.6	11,772	125.3
60	44.1	21.4	19.6	12,003	0.2	20.2	30.51	12,872	127.2

than the ones proposed by B&B algorithm even better about 30 %. Apart from better quality of the solution itself proposed by ACO algorithm, we should notice that the total quality of the system is also very high.

3.2 Power Consumption and System Cost Optimization

This is another example of joint optimization, where we look for the cheapest and the most efficient system.

Time, which has passed until solution was found and the parameters of the system are presented in the Table 3. This example illustrates that ACO algorithm

Table 3 Parameters of the target system (optimization 3.2)

Number of tasks	Ant colony				Branch and bound				Quality
	Time	Length	Cost	Power	Time	Length	Cost	Power	%
5	0.3	20.0	2.1	1600	0.5	20.1	2.0	1580	103
10	0.5	30.1	2.2	2000	0.2	30.1	2.0	2001	100.4
15	3.5	30.09	2.0	3000	0.5	4.5	2.11	3094	72.3
20	4.5	18.3	5.2	3780	0.4	6.0	3.09	3799	71.8
25	10.1	22.2	5.1	4732	0.1	7.9	2.11	5282	72.5
30	12.5	27.2	5.5	5670	0.9	10.1	2.78	6396	71.5
35	12.1	21.2	10.1	6677	0.8	11.3	3.51	7448	62.4
40	27.3	28.1	10.9	7869	0.6	13.5	3.7	8609	60.3
45	16.5	33.2	11.7	8835	0.5	15.1	2.9	9614	60.9
50	57.5	32.1	10.9	9673	0.4	16.2	3.4	10,693	60.7
55	43.5	38.1	10.5	10,766	0.4	18.1	3.6	11,772	61.3
60	55.5	37.3	11.5	11,822	0.2	20.2	3.51	12,872	59.2

isn't better than greedy algorithms for all kinds of problems. When the number of tasks grows, the quality of solution decreases more and more, but you cannot say the same about the quality of system; the ACO algorithm shows, that at the higher expenditure you can obtain solution which is economical and fast at the same time.

4 Conclusion

Basing on the above research you may say, that the ACO algorithm is better suitable for both one and multi-objective analyses. The systems obtained (as a result of ACO algorithm) even in the worst case were only insignificantly worse than solutions obtained by B&B algorithm. Furthermore, the use of s-co-synthesis method significantly improved the quality of obtained solutions. In the case of multi-objective synthesis, heuristic algorithm gave comparable results for optimized parameters and at the same time, the final grade of the systems it proposed was much better. The calculative experiments prove the superiority of synergic design over the "old" synthesis and heuristic algorithms over the greedy ones. The heuristic algorithms handle the NP-complete problems [3] much better than the greedy algorithms. It is because they approach the problem in a way that let them pre-analyze the good solutions and immediately start the optimization of bigger number of parameters in the consecutive steps. The solution suggested in the paper may be applied in supporting computer system prototyping, for dependable and fault-tolerant multiprocessors systems and grid system, too. The model presented for s-co-synthesis and the experimental results allow a further research in this area. For example, other heuristics may be applied. One may also specify additional optimality criteria. The above issues are now studied.

References

1. Aggoune, R.: Minimizing the makespan for the flow shop scheduling problem with availability constraints. Eur. J. Oper. Res. **153**, 534–543 (2004)
2. Gajski, D.: Principles of Digital Design. Prentice Hall, Upper Saddle River, NJ (1997)
3. Coffman, Jr., E.G.: Computer and Job-shop scheduling theory. Wiley, New York (1976)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisko (1979)
5. Dick, R.P., Jha, N.K.: COWLS: hardware-software co-synthesis of distributed wireless low-power client-server systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(1), 2–16 (2004)
6. Montgomery, J., Fayad, C., Petrovic, S.: Solution representation for job shop scheduling problems in ant colony optimization. LNCS **4150**, 484–491 (2006)
7. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for discrete optimization. Artif. Life **5**(2), 137–172 (1999)
8. Mitten, L.G.: Branch-and-bound methods: general formulation and properties. Oper. Res. **18**, 24–34 (1970)

Elastic FOPID+FIR Controller Design Using Hybrid Population-Based Algorithm

Krystian Łapa

Abstract In this paper a new method for elastic H_∞-optimal fractional order PID with FIR filters (FOPID + FIR) controller design using hybrid population-based algorithm is presented. With the use of a population-based algorithm an initial structure of the controller is adjusted in a such way that the designed controller fulfills the control objective in the best way possible. Moreover, in the control process the controller feedback signals' noise and discretization were taken into consideration. The goal of this paper is to show the influence of using FIR filters and FOPID controller structure on accuracy and to present possibilities of designing elastic controller structure using proposed hybrid population-based algorithm. The proposed method was tested on typical control problem.

Keywords PID controller FOPID controller · FIR filter Hybrid algorithms

1 Introduction

The problem of designing control systems is well known in the literature [1]. This is due to the fact that the quality of work of individual parts or even of entire machines mainly depends on the characteristics of the used controller. The proper controller design should take following elements under consideration: indication of measurable signals, selection of the controller structure, tuning of controller parameters and implementation in target hardware platform with fulfillment of requirements of real-time work. Usually these steps are performed in the presented order.

In the literature there are well-known controller structures such as: controllers structures based on the combination of linear correction terms, e.g. PID controllers (optionally with gain scheduling algorithm, with feed-forward path or with additional low-pass filters [2]), state feedback controllers, nonlinear controllers based on

K. Łapa (✉)
Institute of Computational Intelligence, Częstochowa University of Technology, Częstochowa, Poland
e-mail: krystian.lapa@iisi.pcz.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_2

computational intelligence and hybrid controllers, in which are combined approaches from other groups. However, in practice PID controllers are used the most often [1]. It is a result of widespread knowledge of how they work and their relatively simple implementation in a microprocessor-based control systems. An extension of PID controllers are, among the others, H∞-optimal (in these methods the problem of control is defined as the optimal control task, and then the controller, which may perform such a task, is designed) Fractional Order PID (FOPID) controllers. These controllers improve responses with respect to rational-type controllers such as PID [4].

The problems associated with designing of the controller apply, among the others, to: difficult and time consuming design process, modification of the structure to support more than one feedback signal (typical PID controllers consist of single PID block for processing a single controller signal), proper selection of the structure parameters, noise reduction, not taking into considering discretization of the feedback signals, etc.

Among the experimental methods for the design of control systems, methods based on artificial intelligence [3] and in particular the methods of evolution [5] are becoming more common. The methods of evolution are based on populations of the solutions, where each solution can represent the structure and the parameters of the single controller [8]. During evolution the population can improve (better solutions are being find) by modifying or mixing system structure and parameters between solutions. This process is usually based on fitness function value calculated for each solution.

In this paper a new method for elastic H∞-optimal fractional order PID with FIR filters (FOPID + FIR) controller design using hybrid population-based algorithm is presented. Low-pass finite-impulse-response filters (FIR) with programmable characteristics for each of the measurement signals are used in the feedback loop. These filters are designed to suppress interference that could disrupt the work of the control system. Due to that it is possible to find the structure and parameters of the controller, which makes it immune to this type of interference. From the other hand, using proposed elastic FOPID controller instead of standard PID controller allow to handle higher order processes by performing optimization with various integral performance indices. In the design process an universal initial structure is proposed, which in the process of evolution will be adjusted in a way that the designed controller fulfills the control objective in the best way possible. This elastic structure consists of FOPID functional blocks and FIR filters, both with programmable structures, connections and parameters. Due to this approach the design of the control system can be regarded as one continuous process, unlike the commonly used method of trial and error. As a result, the process of controller design is performed easier and faster. Details of the proposed method are described in Sect. 3.

This paper is organized into 5 sections. Section 2 contains a description of the elastic FOPID + FIR controller structure, while Sect. 3 shows the proposed evolutionary algorithm used to design control system. Simulation results are presented in Sect. 4. Conclusions are drawn in Sect. 5.

2 Description of Proposed FOPID + FIR Controller

Proposed controller is based on elastic structure, which among the others, depends from number of controller input signals fb_i, $i = 1, \ldots, FB$, FB stands for number of feedback signals (see Fig. 1). In proposed structure assumptions that fb_1 stands for desired value of fb_2 and the rest of the feedback signals stand for additional measurable signals are stated. Moreover, FOPID elements and FIR filters and their inner elements can be dynamically switched off or on by changing controller parameters. Due to that, the design of the controller should not only consider selecting the real parameters of the controller but also integer parameters encoding its structure. The typical FOPID control block consist of five elements: proportional P, integral I and λ and differential D and μ and its output is calculated as follows:

$$u(t) = K^{\mathrm{P}} e(t) + K^{\mathrm{I}} \left(\int_0^t e(t)dt \right)^{-\lambda} + K^{\mathrm{D}} \left(\frac{de(t)}{dt} \right)^{\mu}, \tag{1}$$

where K^{P}, K^{I} and K^{D} stand respectively for parameters of P, I and D elements of control block, λ and μ are additional degrees of freedom in a comparison to typical PID controller structure, $e(t)$ stands for input of FOPID block. These parameters allow to handle higher order processes by performing optimization with various integral performance indices. The proposed elastic FOPID structure (noted as Control Block CB—see Fig. 2a) allows for additional reduction of P, I, D, λ and μ elements by using integer values C^{P}, C^{I}, C^{D}, C^{λ}, C^{μ} and reduction of whole control block by using integer value C^{CB}. The reduction takes place if the integer values are set to 0. Then, the output of the proposed FOPID takes the following form:

$$u(t) = \begin{cases} C^{\mathrm{P}} K^{\mathrm{P}} e(t) + C^{\mathrm{I}} K^{\mathrm{I}} \left(\int_0^t e(t)dt \right)^{ka(t)} + C^{\mathrm{D}} K^{\mathrm{D}} \left(\frac{de(t)}{dt} \right)^{kb(t)} & \text{for} \quad C^{\mathrm{CB}} = 1 \\ e(t) & \text{for} \quad C^{\mathrm{CB}} = 0 \end{cases}, \tag{2}$$

where $ka(t)$ stands for $-\lambda$ when $C^{\lambda} = 1$ and 1 when $C^{\lambda} = 0$, $kb(t)$ stands for μ when $C^{\mu} = 1$ and 1 when $C^{\mu} = 0$ (if $C^{\lambda} = 0$ and $C^{\mu} = 0$ proposed FOPID controller work

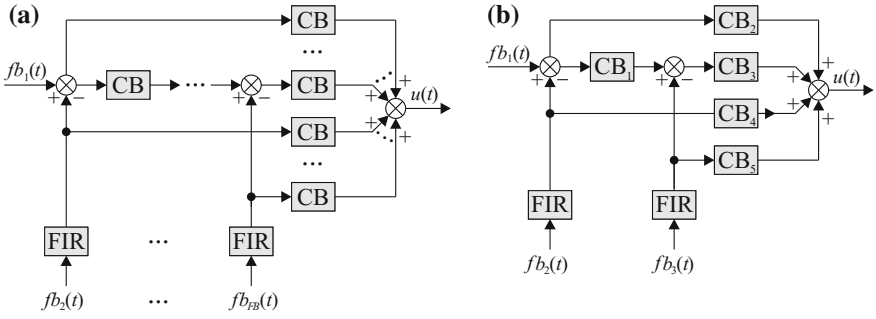


Fig. 1 Proposed controller structure: **a** with any number of *FB* feedback signals, **b** with 3 feedback signals

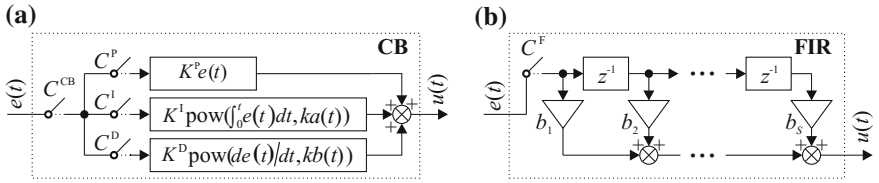


Fig. 2 Structure of: **a** proposed elastic control block CB based on FOPID, **b** proposed elastic filter FIR, z^{-1} stands for values from previous time step

as typical PID controller). The proposed FIR filters used in controller are based on typical FIR filters (see Fig. 2b) with using an additional integer parameter C^{F} standing for reduction of the filter. Thus, the output of the proposed filter takes the following form:

$$u(t) = \begin{cases} \sum_{s=1}^{S} b_s e(t-s-1) & \text{for} \quad C^{\mathrm{F}} = 1 \\ e(t) & \text{for} \quad C^{\mathrm{F}} = 0 \end{cases}, \tag{3}$$

where $e(t)$ stands for input value, $e(t-i)$ stands for input value from $t-i$ time step, b_s stands for weights of filter, $s = 1, \ldots, S$, S stands for length of the filter (S has to be an odd number). The weights of the filter are calculated using filter parameters: transition frequency ft and length of the filter S, which are a part of the elastic structure of the controller and should be selected by learning algorithm as well. The weights values b_s are calculated as follows:

$$b_s = \begin{cases} \frac{\sin\left(2\pi ft\left|s-\frac{1}{2}(S-1)\right|\right)}{\pi\left|s-\frac{1}{2}(S-1)\right|} & \text{for} \quad s = \frac{1}{2}(S-1) \\ 2ft & \text{for} \quad s \neq \frac{1}{2}(S-1) \end{cases}. \tag{4}$$

The proposed controller structure is characterized by the following advantages: (a) possibilities of processing any number of feedback signals fb_i, (b) it uses cascade control blocks configuration which allows us to obtain good accuracy of the controller, (c) the structure is dynamic, each CB block elements (P, I, D, λ, μ) and filter FIR can be switched off or on, (d) it has great capabilities of learning due to many selectable parameters (e) it is able to minimize the impact of feedback signals noise by use of the FIR filters.

3 Description of the Proposed Hybrid Algorithm

In our paper a hybrid evolutionary algorithm is used to select the proposed controller parameters and structure. It is based on an ensemble of genetic algorithm (to select controller structure) and evolutionary strategy (to select controller

parameters). This ensemble was proposed in our previous work and it achieved good results. In this paper we propose a number of improvements that may allow us to obtain better performance. These improvements consider, most of all, iteration-dependent parameters of learning process and their description can be found in detail in the current section.

3.1 Encoding of the Controller Parameters

The parameters and the structure of the proposed controller are encoded in chromosome $\mathbf{X}_{ch}$ defined as follows:

$$\mathbf{X}_{ch} = \left\{\mathbf{X}_{ch}^{\text{par}}, \mathbf{X}_{ch}^{\text{str}}\right\}, \tag{5}$$

where part $\mathbf{X}_{ch}^{\text{par}}$ encodes the real parameters of the controller and part $\mathbf{X}_{ch}^{\text{str}}$ encodes integer parameters of the controller. The part $\mathbf{X}_{ch}^{\text{par}}$ is defined as follows:

$$\mathbf{X}_{ch}^{\text{par}} = \left\{ \begin{array}{c} K_1^{\text{P}}, K_1^{\text{I}}, K_1^{\text{D}}, \lambda_1, \mu_1 \ldots, \\ K_M^{\text{P}}, K_M^{\text{I}}, K_M^{\text{D}}, \lambda_M, \mu_M \\ ft_1, \ldots ft_R \end{array} \right\} = \left\{X_{ch,1}^{\text{par}}, \ldots, X_{ch,L^{\text{par}}}^{\text{par}}\right\}, \tag{6}$$

where $K_m^{\text{P}} \in [0, 20], K_m^{\text{I}} \in [0, 50], K_m^{\text{D}} \in [0, 5], \lambda_m \in [0.5, 2.0], \mu_m \in [0.5, 2.0]$, stand for CB P, I, D, λ, μ parameters, $m = 1, \ldots, M$, M stands for number of CB blocks, $ft_r \in [0.1, 0.5]$ stands for transition frequency, $r = 1, \ldots, R$, $R = FB - 1$ stands for number of filters, $L^{\text{par}} = 5M + R$ stands for number of genes in part $\mathbf{X}_{ch}^{\text{par}}$. The part $\mathbf{X}_{ch}^{\text{str}}$ is defined as follows:

$$\mathbf{X}_{ch}^{\text{str}} = \left\{ \begin{array}{c} C_1^{\text{P}}, C_1^{\text{I}}, C_1^{\text{D}}, C_1^{\lambda}, C_1^{\mu}, \ldots, \\ C_M^{\text{P}}, C_M^{\text{I}}, C_M^{\text{D}}, C_M^{\lambda}, C_M^{\mu}, \\ C_1^{\text{CB}}, \ldots, C_M^{\text{CB}}, C_1^{\text{F}}, \ldots, C_R^{\text{F}} \\ F_1, \ldots, F_R \end{array} \right\} = \left\{X_{ch,1}^{\text{str}}, \ldots, X_{ch,L^{\text{str}}}^{\text{str}}\right\}, \tag{7}$$

where $C_m^{\text{P}} \in \{0, 1\}, C_m^{\text{I}} \in \{0, 1\}, C_m^{\text{D}} \in \{0, 1\}, C_m^{\lambda} \in \{0, 1\}, C_m^{\mu} \in \{0, 1\}$ stand for activation of CB P, I, D, λ, μ elements (values equal to 1 stands for active element), $C_m^{\text{CB}} \in \{0, 1\}$ stands for activation of mth control block, $C_r^{\text{F}} \in \{0, 1\}$ stands for activation of rth filter (values equal to 1 stands for active element), $F_r \in \{0, \ldots, 9\}$ stands for length of the filter (real length of the filter is calculated as $S_r = 5 + 2F_r$ to obtain at least 5-size long filters), $L^{\text{str}} = 6M + 2R$ stands for number of genes in part $\mathbf{X}_{ch}^{\text{str}}$.

3.2 Proposed Algorithm Description

Proposed algorithm is based on new iteration-dependent mutation and crossover from genetic algorithm and evolutionary strategy. The algorithm works according to the following steps

- **Step 1. Initialization**. In this step the value *iteration* is set to 0. Next the N individuals (each individual $\mathbf{X}_{ch}$ represents controller encoded by chromosome (5) are randomly initialized and stored in population $\mathbf{P}$. The initialization of individuals' genes is realized as follows: $X_{ch,g}^{\text{par}} = U^{\text{g}}(\underline{X}_{ch,g}^{\text{par}}, \bar{X}_{ch,g}^{\text{par}})$, where $U^{\text{g}}(a, b)$ returns a random real value from the range $[a, b]$, $\underline{X}_{ch,g}^{\text{par}}$ and $\bar{X}_{ch,g}^{\text{par}}$ stand respectively for minims and maxims values of genes $X_{ch,g}^{\text{par}}$, $g = 1, \ldots, L^{\text{par}}$, $X_{ch,h}^{\text{str}} = U^{\text{h}}(\underline{X}_{ch,h}^{\text{str}}, \bar{X}_{ch,h}^{\text{str}})$, where $U^{\text{h}}(a, b)$ returns random integer value from the range $[a, b]$. $\underline{X}_{ch,h}^{\text{str}}$ and $\bar{X}_{ch,h}^{\text{str}}$ stand respectively for minims and maxims values of genes $X_{ch,h}^{\text{str}}$,$h = 1, \ldots, L^{\text{str}}$.
- **Step 2. Evaluation**. In this step each individual is evaluated by fitness function defined as follows:

$$\text{ff}(\mathbf{X}_{ch}) = \sum_{f=1}^{F} \text{w}_f \cdot \text{ffcom}_f(\mathbf{X}_{ch}), \tag{8}$$

  where $\text{ffcom}_f(\mathbf{X}_{ch})$ stands for fitness function components which depend from simulation problem (see Sect. 4), w_f stands for weights of components, $f = 1, \ldots, F$, F stands for number of fitness function components.
- **Step 3. Probabilities calculation**. In this step the value *iteration* is incremented. Next, the dynamic parameters for mutation and crossover are calculated as follows: individual mutation probability $p_1 = 0.10 + 0.20 \cdot \alpha$, gene mutation range $p_2 = 0.05 + 0.20 \cdot \alpha$, gene mutation probability $p_3 = 0.01 + 0.10 \cdot \alpha$, α stands for iteration-dependent value calculated as:

$$\alpha = 1 - \frac{iteration}{iteration^{\max}}. \tag{9}$$

  where $iteration^{\max}$ stands for maximum number of algorithm iterations. The purpose of iteration dependent probabilities is to increase the possibilities of accurate exploration of space exploration by decreasing influence and range of mutation.
- **Step 4. Reproduction.** In this step a N new individuals are created and stored in population $\mathbf{P}'$. For each individual the condition $U^{\text{g}}(0, 1) < p_c$ is checked (where $p_c \in (0, 1)$ stands for crossover probability). If this condition is met, new individual is created as a result of crossover between two individuals selected by the roulette wheel method [7] from population $\mathbf{P}$. Otherwise, the individual is created as a result of cloning and mutating of one individual, which

is also selected by the roulette wheel method [7] from population **P**. The mutation is performed according to Eq. (11) (see Step 5). The genes obtained from crossover are calculated as:

$$\begin{cases} X_{ch,g}^{\mathrm{par}} = \begin{cases} X_{ch,g}^{\mathrm{A,par}} & \text{for} \quad U^{\mathrm{g}}(0,1) < 0.5 \quad \text{and} \quad U^{\mathrm{g}}(0,1) < p_3 \\ X_{ch,g}^{\mathrm{B,par}} & \text{for} \quad U^{\mathrm{g}}(0,1) \geq 0.5 \quad \text{and} \quad U^{\mathrm{g}}(0,1) < p_3 \\ X_{ch,g}^{\mathrm{A,par}} + U^{\mathrm{g}}(0,1) \cdot \left(X_{ch,g}^{\mathrm{B,par}} - X_{ch,g}^{\mathrm{A,par}} \right) & \text{for} \quad U^{\mathrm{g}}(0,1) \geq p_3 \end{cases} \\ X_{ch,h}^{\mathrm{str}} = \begin{cases} X_{ch,h}^{\mathrm{A,str}} & \text{for} \quad U^{\mathrm{g}}(0,1) < 0.5 \quad \text{and} \quad U^{\mathrm{g}}(0,1) < p_3 \\ X_{ch,h}^{\mathrm{B,str}} & \text{for} \quad U^{\mathrm{g}}(0,1) \geq 0.5 \quad \text{and} \quad U^{\mathrm{g}}(0,1) < p_3 \\ X_{ch,h}^{\mathrm{A,str}} + U^{\mathrm{h}}\left(X_{ch,h}^{\mathrm{A,str}}, X_{ch,h}^{\mathrm{B,str}} \right) & \text{for} \quad U^{\mathrm{g}}(0,1) \geq p_3 \end{cases} \end{cases}, \tag{10}$$

where $X_{ch,g/h}^{\mathrm{A,str/par}}$ and $X_{ch,g/h}^{\mathrm{B,str/par}}$ stand respectively for genes from the first and second parent. The purpose of Eq. (10) is to increase chance to select gene values directly from parents or in the other case to select gene values between gene values of parents (if condition $U^{\mathrm{g}}(0,1) \geq p_3$ is met).

- **Step 5. Mutation**. In this step genes of individuals from population $\mathbf{P}'$ are mutated. For each individual the condition $U^{\mathrm{g}}(0,1) < p_m$ is checked (where p_m stands for mutation probability). If this condition is met, genes of $\mathbf{X}_{ch}$ are modified as follows:

$$\begin{cases} X_{ch,g}^{\mathrm{par}} = \begin{cases} X_{ch,g}^{\mathrm{par}} + U^{g}(-1,1) \cdot p_2 \cdot \left(\bar{X}_{ch,g}^{\mathrm{par}} - \underline{X}_{ch,g}^{\mathrm{par}} \right) & \text{for} \quad U^{\mathrm{g}}(0,1) < p_1 \\ X_{ch,g}^{\mathrm{par}} & \text{for} \quad U^{\mathrm{g}}(0,1) \geq p_1 \end{cases} \\ X_{ch,h}^{\mathrm{str}} = \begin{cases} X_{ch,h}^{\mathrm{str}} + U^{h}(-1,1) & \text{for} \quad U^{\mathrm{g}}(0,1) < p_3 \\ X_{ch,h}^{\mathrm{str}} & \text{for} \quad U^{\mathrm{g}}(0,1) \geq p_3 \end{cases} \end{cases}. \tag{11}$$

- **Step 6. Repair**. This step purpose is to repair (cut to specified ranged) gene values of individuals from population $\mathbf{P}'$, which is executed as follows:

$$\begin{cases} X_{ch,g}^{\mathrm{par}} = \min\left(\bar{X}_{ch,g}^{\mathrm{par}}, \max\left(\underline{X}_{ch,g}^{\mathrm{par}}, X_{ch,g}^{\mathrm{par}} \right) \right) \\ X_{ch,h}^{\mathrm{str}} = \min\left(\bar{X}_{ch,h}^{\mathrm{str}}, \max\left(\underline{X}_{ch,h}^{\mathrm{str}}, X_{ch,h}^{\mathrm{str}} \right) \right) \end{cases}. \tag{12}$$

- **Step 7. Evaluation**. In this step all individuals from population $\mathbf{P}'$ are evaluated according to fitness function (8).
- **Step 8. Merging**. This step aim is to select the best N individuals from merged populations **P** and $\mathbf{P}'$. Selected individuals replace population **P**.
- **Step 9**. **Stopping condition**. In this step the stop condition is checked

- ($iteration \geq iteration^{\max}$). If this condition is met, algorithm stops and the best individual according to the fitness function value is presented. Otherwise, the algorithm goes back to Step 3.

4 Simulation Results

In our simulations a problem of designing controller structure and tuning parameters for double spring-mass-damp object was considered (see Fig. 3). More details about this model can be found in our previous paper [9]. Object parameters were set as follows: spring constant $k = 10\,\text{N/m}$, coefficient of friction $\mu = 0.5$, masses $m_1 = m_2 = 0.2\,\text{kg}$. Initial values of: s^1, v^1, s^2, v^2 (s stands for position, v stands for velocity) were set to zero, and s^* is a desired position of mass m_1 (see Fig. 3), simulation length T^{all} was set to 10 s, output signal of the controller was limited to the range $u \in (-2, +2)$, quantization resolution for the output signal of the controller and for the position sensor for s^1 and s^2 was set to 8 bit, noise level of feedback signals was set to 1 %, time step in the simulation was equal to $T = 0.1\,\text{ms}$, while interval between subsequent controller activations were set to 20 simulation steps, number of model iteration is calculated as $Z = T^{\text{all}}/T$. The feedback signals for the controller was chosen as: $fb_1 = s^*$, $fb_2 = s^1$, $fb_3 = s^2$.

4.1 Problem Evaluation

For problem under consideration a trapezoidal shape of desired signal s^* was used (see Fig. 4). Moreover, a following fitness function components (8) were used (additionally, a settling time can be included in a further research):

- Complexity of the controller:

$$\text{ffcom}_1(\mathbf{X}_{ch}) = \frac{1}{L^{\text{str}}} \sum_{g=1}^{L^{\text{str}}} \mathbf{X}_{ch,g}^{\text{str}}, \tag{13}$$

- *RMSE* standing for accuracy of the controlled object:

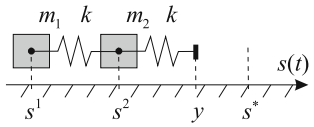


Fig. 3 Simulated spring-mass-damp object

$$\mathrm{ffcom}_2(\mathbf{X_{ch}}) = RMSE = \sqrt{\frac{1}{Z}\cdot\sum_{i=1}^{Z}\varepsilon_i^2} = \sqrt{\frac{1}{Z}\cdot\sum_{i=1}^{Z}\left(s_i^* - s_i^1\right)^2}, \tag{14}$$

- Overshooting of the controller:

$$\mathrm{ffcom}_3(\mathbf{X_{ch}}) = \max_{i=1,\ldots,Z}\left\{s_i^1\right\}. \tag{15}$$

- Oscillations of the output of the controller:

$$\mathrm{ffcom}_4(\mathbf{X_{ch}}) = \sum_{o=1}^{O-1}\sqrt{|r_o - r_{o+1}|}, \tag{16}$$

where r_o stands for each local minims and maxims of the output values of the controller (minims and maxims were selected with ignoring noise influence on the signals), $o = 1, \ldots, O$, O stands for number of minims and maxims of oscillations. The aim of the Eq. (16) is to promote solutions with low number of low height oscillations of the controller.

4.2 Simulation Parameters

In the simulations the following values of parameters were set experimentally: fitness function components weights $w_1 = 0.1$, $w_2 = 10$, $w_3 = 0.01$, $w_4 = 0.1$, crossover probability $p_c = 0.75$, mutation probability $p_m = 0.75$, number of algorithm iterations $iteration^{\max} = 1000$, number of individuals in populations $N = 100$. In the simulations four cases presented in Table 1 were tested to show the effectiveness of the proposed controller and learning algorithm (case 1 corresponds to solution presented in [9]). For each case simulations were repeat 100 times and results were averaged.

4.3 Obtained Results

The averaged simulations results are presented in Table 2, the best simulations results are presented in Table 3, Figs. 4 and 5.

Table 1 Simulation cases

Case	Controller type	Filters	λ, μ always active
1	PID (FOPID with $\lambda = -1, \mu = 1$)	no	no
2	PID (FOPID + FIR with $\lambda = -1, \mu = 1$)	yes	no
3	proposed elastic FOPID + FIR	yes	no
4	proposed elastic FOPID + FIR	yes	yes

Table 2 Averaged simulation results

Case	ff(·)	$ffcom_1(\cdot)$ complexity	$ffcom_2(\cdot)$ accuracy	$ffcom_3(\cdot)$ oscillations	$ffcom_4(\cdot)$ overshooting
1	3.628	**0.437**	0.162	31.743	1.190
2	3.080	0.502	0.144	**13.767**	**1.160**
3	2.382	0.640	0.114	14.468	1.195
4	**2.373**	0.664	**0.107**	27.651	1.221

Table 3 Best simulation results

Case	ff(·)	$ffcom_1(\cdot)$ complexity	$ffcom_2(\cdot)$ accuracy	$ffcom_3(\cdot)$ oscillations	$ffcom_4(\cdot)$ overshooting
1	2.438	**0.467**	0.103	28.572	1.294
2	1.807	0.533	0.076	21.885	1.157
3	1.705	0.680	0.074	26.230	1.155
4	**1.307**	0.840	**0.059**	**19.689**	**1.062**

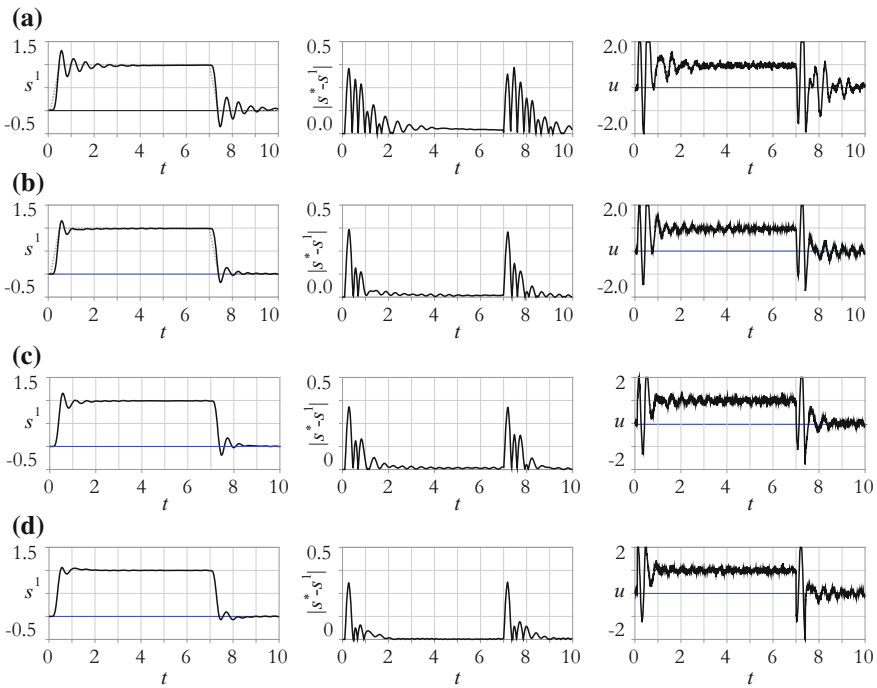


Fig. 4 Best simulations results for: **a** case 1, **b** case 2, **c** case 3, **d** case 4. s^1 stands for position of the mass m_1, $|s^* - s^1|$ stands for difference with desired position of mass m_1, u stands for output of the controller

4.4 Simulation Results

Conclusions from the simulation are as follows: (a) adding filters allowed for reduce impact of feedback signals noise (see Fig. 4a), Fig. 4b) and ff$(\cdot)$ values in Table 2); (b) proposed elastic FOPID structure allowed for increase controller accuracy by around 20 % with increase of controller complexity by around 10 % (see Fig. 4b, c) and ffcom$_1(\cdot)$, ffcom$_2(\cdot)$ values in Table 2); (c) using static active λ, μ elements in FOPID structure allowed for next 5 % improvement in controller accuracy with slighter increase in controller complexity (see Fig. 4c, d) and ffcom$_1(\cdot)$, ffcom$_2(\cdot)$ values in Table 2); (d) the best obtained accuracy of the controller (see Table 3—case 4) is better than accuracy obtained by hybrid multi-population algorithms without noise of the signals under consideration (see our previous work [6]); (e) the obtained controller structures are simple and clear (see Fig. 5).

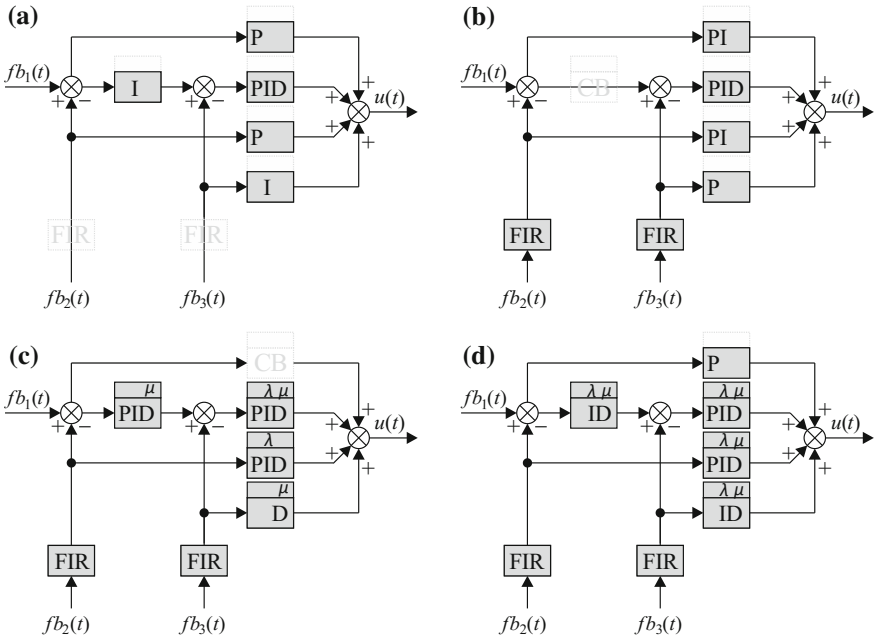


Fig. 5 Best simulations structures obtained for: **a** case 1, **b** case 2, **c** case 3, **d** case 4. Gray rectangles stands for reduced elements of the controller

5 Conclusions

In this paper a new method for elastic H_∞-optimal fractional order PID with FIR filters (FOPID + FIR) controller design using hybrid population-based algorithm was proposed. The proposed elastic structure of the controller (FOPID + FIR) allowed to obtain overall good controllers (with good accuracy, small number of oscillations, low overshooting) with taking under consideration feedback signal noise and discretization. Moreover, the proposed training algorithm, which allows reduction of any component of the controller and simultaneously selection of its parameters, allowed to obtain a very good results in terms of accuracy. It can be said that the proposed elastic FOPID + FIR controller is superior in comparison to typical PID/FOPID controllers.

Acknowledgment The project was financed by the National Science Centre (Poland) on the basis of the decision number DEC-2012/05/B/ST7/02138.

References

1. Alia, M.A.K., Younes, T.M., Alsabbah, S.A.: A design of a PID self-tuning controller using LabVIEW. J. Softw. Eng. Appl. **4**, 161–171 (2011)
2. Astrom, K.J., Hagglund, T.: PID Controllers: Theory, Design, and Tuning. Instrument Society of America: Research Triangle Park, book (1995)
3. Cpałka, K., Łapa, K., Przybył, A.: A new approach to design of control systems using genetic programming. Inf. Technol. Control **44**(4), 433–442 (2015)
4. Ho, S.J., Ho, S.Y., Shu, L.S.: OSA: Orthogonal simulated annealing algorithm and its application to designing mixed H2 = H1 optimal controllers. IEEE Trans. Syst. Man Cybern. **34**(5), 588–600 (2004)
5. Łapa, K., Przybył, A., Cpałka, K.: A new approach to designing interpretable models of dynamic systems. Lect. Notes Artif. Intell. **7895**, 523–534 (2013)
6. Łapa, K., Szczypta, J., Venkatesan, R.: Aspects of structure and parameters selection of control systems using selected multi-population algorithms. Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, vol. 9120, pp. 247–260 (2015)
7. Rutkowski, L.: Computational Intelligence. Springer (2008)
8. Szczypta, J., Łapa, K., Shao, Z.: Aspects of the selection of the structure and parameters of controllers using selected population based algorithms. In: Artificial Intelligence and Soft Computing. Springer International Publishing, pp. 440–454 (2014)
9. Szczypta, J., Przybył, A., Cpałka, K.: Some aspects of evolutionary designing optimal controllers. Artif. Intell. Soft Comput. part II **7895**, 91–100 (2013)

Optimization of Read-Only Memory Program Models Mapping into the FPGA Architecture

Viktor Melnyk and Ivan Lopit

Abstract The problem of mapping the read-only memory program models, being the components of the application-specific processors, into the programmable logical integral circuit architecture will be considered. The existing approaches to the data compression in the read-only memory devices are analyzed and, alternatively, a new approach is suggested allowing a high degree of compression to be achieved and the FPGA resources to be used more rationally.

Keywords FPGA · Application-specific processors program models · Read-only memory · Data compression

1 Introduction

Given a modern state of computer component base, the computer system performance is being growing mostly extensively, i.e. in a way of increasing the number of general-purpose processor cores and their operating frequency. At the same time, the approach of using general-purpose processors to achieve high performance indices has principal shortcomings. The main of them are high power consumption and low efficiency of hardware use. To overcome the above shortcomings the computer systems are created with application-specific processors, however, they are efficient for the narrow classes of algorithms only, whereas their development requires substantial efforts and resources. One of the variants of solving this problem relates to generating the application-specific processor program models (ASPPM) from a high-level presentation of their operation algorithm and their implementation in the field-programmable gate arrays (FPGA). This results in the

V. Melnyk (✉)
The John Paul II Catholic University of Lublin, Lublin, Poland
e-mail: vmelnyk@kul.pl

V. Melnyk · I. Lopit
Lviv Polytechnic National University, Lviv, Ukraine
e-mail: lopit.i.i@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_3

application-specific processors used as the computer system accelerators with functions varied by the FPGA reconfiguration and creation of another application-specific processor (ASP) in it.

One of the frequently used ASPPM components is the read-only memory (ROM). It is used for storing constant values and commands, implementing the tabular and tabular-algorithmic computing devices etc. One of the problem issues in realizing ROM in FPGA is not rational use of its resources related to the excessive character of data representation. Compression of the data contained in ROM is one of the possible solutions of this problem.

Data compression is a set of rules for presenting the above data in another form, in which they occupy the less volume. In fact, data compression means their recoding. The lossless data compression algorithms are divided into two principal classes: data coding using a dictionary and entropic data coding.

The most widely used algorithms in the dictionary-related coding class are the Lempel–Ziv–Welch (LZW) one [1] and its variations, while in the entropic coding class these are the Huffman code-based ones [2].

2 Research and Publications Survey

The Lempel–Ziv–Welch algorithms are described in [1], whereas the Huffman ones —in [2]. In [3], implementation of the data decompression function in FPGA through the embedded memory units according to the LZW algorithm is described. The hardware Huffman decoder implemented in FPGA is described in [4].

3 Problem Stating

Modern compression algorithms described in [1, 2] demonstrate high compression ratio, but their hardware implementation complexity and low speed restrict their wide use for the data compression in ROM. The analysis of the variants of decoder hardware implementation [3, 4] has shown a series of shortcomings in their characteristics, namely:

- low operating clock frequency;
- significant delay with access to the next memory cell: for [3]—6 cycles, for [4]—4 cycles of synchronization signal.

The above shortcomings affect essentially the efficiency of ASPPM with ROM components. Searching for solutions free from the above disadvantages was the goal of study carried out in this paper.

4 Algorithm of Optimizing the ROM Program Model Mapping into the FPGA Architecture by the Data Group Compression

When generating ASPPM from the high-level programming language, the ROM content is usually foreknown. Therefore, the generating system may produce any necessary content transformations at the moment of the program model synthesizing. Data rearrangement is one of the simplest data transformations in the hardware implementation. It requires no hardware expenses and could be done by changing the order of port connecting. Using the above specific features, the authors have suggested the data group compression algorithm that uses the dictionary method. This algorithm could be conditionally divided into two parts described below, i.e. compression by methods simple to be hardware implemented and search for the optimal data group these methods are executed over.

4.1 Data Compression by the Dictionary Method

Data compression method using dictionary includes the fixed-length value set coding into the coded value set and the table of keys for their recovery. Figure 1 illustrates schematically the principle of the hardware implementation of this method.

The operation principle is as follows. First, the access to key takes place, this key enables the access to the unique value. The time of waiting for new value is 2 synchronization cycles.

The compression ratio for this method could be calculated by the following formula:

$$\eta = \frac{[\log_2 u] \cdot h + u \cdot w}{h \cdot w}, \tag{1}$$

where u is a number of unique values in the primary table, w is the word length of the data in this table, h is its height.

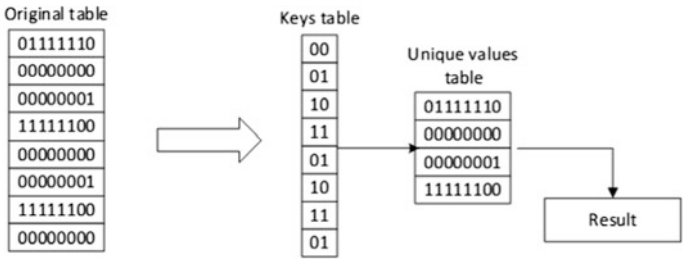


Fig. 1 Principle of the hardware implementation of data compression by the dictionary method

This method is characterized by a relatively low compression degree, but its hardware implementation is simple and ensures high speed.

4.2 Data Compression by the Sequence Length Coding Method

Sequence length coding is the data compression algorithm that replaces the repetitive symbols (data blocks) by the repetitive symbol (data block) and by the number of its repetitions. The principle of the hardware implementation of this method is schematically shown in Fig. 2.

The operation principle of this device is as follows. The index counter contains a current address and the content is read from the sequences table and the values table according to the above address value. The content received from the sequences table comes to the SC counter. When the content of all the SC counter cells becomes 0, the index value is increased by 1 and the new values are read from the sequences and values tables.

The compression coefficient for this method may be calculated as:

$$\eta = \frac{[\log_2 S_{\max}] \cdot us \cdot + w \cdot us}{h \cdot w}, \tag{2}$$

where S_{max} is the largest sequence length, us is the number of unique sequences in the original table, w is the original table capacity, h is the original table height.

One of the shortcomings of this method is the possible misbalance of the sequences table. Such situation takes place when one of the sequences is much longer than the other one and this is illustrated in Fig. 3.

The above disadvantage could be eliminated by the preliminary balancing of the sequences table. Here under balancing we mean a process of searching such representation of sequence at which the size of the table is minimal.

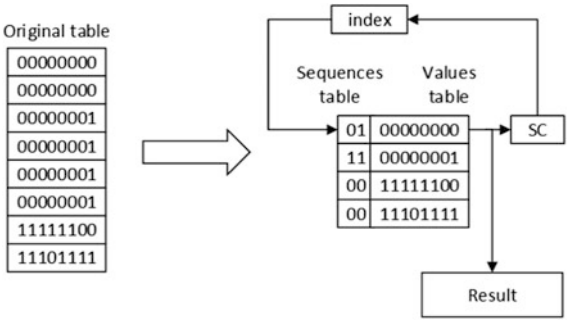


Fig. 2 Principle of the hardware implementation of data compression by the sequence length coding method

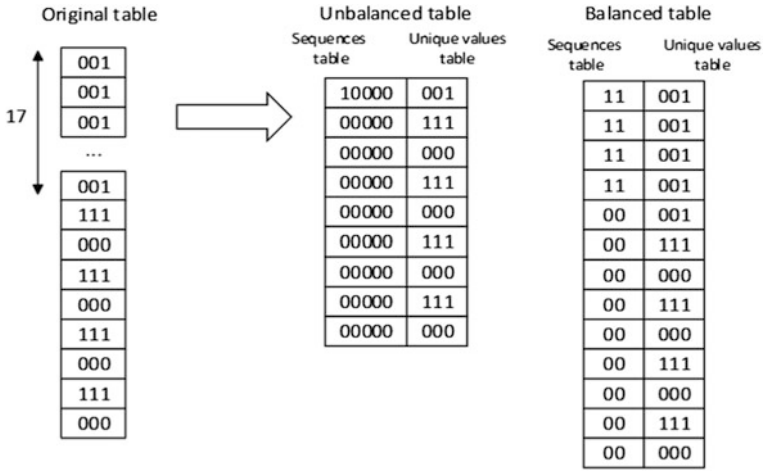


Fig. 3 Example of misbalanced sequences table

4.3 *Data Compression by the Combined Method of Sequence Length and Dictionary Coding*

This method is the combination of the above data compression methods, i.e. the dictionary and the sequence length coding methods. According to it, first the set of the fixed-length values is coded into the coded values set and the key table for their recovery. Then the repeated key sequence lengths are coded. Figure 4 illustrates schematically the principle of this method hardware implementation.

The operation principle is as follows. The index counter contains a current address and the content is read from the sequences table and the keys table according to the above address value. The content received from the sequences

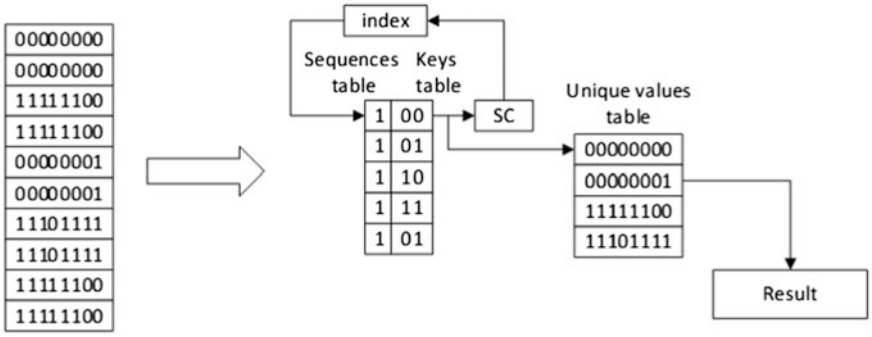


Fig. 4 Principle of the hardware implementation of the data compression by the combined method of sequence length and dictionary coding

table comes to the SC counter. The content received from the keys table is used to receive the value from the unique values table, and this is a result. When the content of all the SC counter cells becomes 0, the index value is increased by 1 and the new values are read from the sequences and keys tables.

The compression ratio for this method could be calculated by the following formula:

$$\eta = \frac{[\log_2 S_{\max}] \cdot us + [\log_2 u] \cdot us + u \cdot w}{h \cdot w}, \tag{3}$$

where S_{max} is the longest sequence length, us is the number of unique sequence in the original table, u is the number of unique values in the original table, w is the original table capacity, h is the original table height.

4.4 Data Grouping Algorithm

Data grouping algorithm includes searching the optimal groups with the use of rearrangements. The rearrangement unit is the column in the memory unit.

The algorithm input data are: memory unit content for compression; w is the original memory unit capacity; h is the number of cells in the memory unit, wc is the subunit capacity after compression, max_c is the maximal value of possible combinations per step; min_c is the minimal value of possible combinations per step.

The output algorithm data is: a set of the memory subunits after compression.

The algorithm criteria are: "speed"—the compression speed is preferred, "compression"—the maximum compression ratio is preferred.

This algorithm includes execution of the following steps:

1. If w/wc is not integer, for compression add the last column to the original memory unit.
2. Create a set of columns R on the basis of the original memory unit.
3. Assign the maximally possible number to the variable-result Gor.
4. Generate a set of substeps S. If the number of $C_w^{wc} > max\_c$, then expand wc into the simple factors that may create a set S, otherwise, w becomes a single element of the set S.
5. For each s from the set S:

   (a) If the number of $C_R^S > min\_c$, choose the next simple factor and multiply it by the existing one. Go to step 5.
   (b) Using C_R^S form all the possible combinations of the group placement—P.
   (c) On the basis of combination of P choose the columns with the relevant indices from R and form a set of possible groups G.
   (d) For each group g in G execute the compression algorithm.

(e) Search for the optimal group *Go* from the set *G* obtained at the step 5 (d). If the optimization criterion is "speed", use the greedy algorithm of search (step 5 (f)). If the optimization criterion is "compression", use the expanded greedy algorithm of search (step 5 (g)).
(f) Greedy algorithm of search. Until the number of groups reaches *R/wc,* search for the minimal group, for which the following equality holds true: $g \cap Go = \emptyset$.
(g) Expanded greedy algorithm of search. For each group *g* in the set *G* execute greedy search (step 5 (f)) with the following exception: instead of the first iteration the minimal element should be the current element *g*.

6. If the size of *Go* obtained from step 5 (f) 5(g) is less than *Gor*, assign *Go* for *Gor*.
7. Form a new set of elements-columns *R*, taking as a basis the set *Gor* obtained in step 5 (f) or 5(g).
8. End.

This algorithm includes iteration search for the optimal placement of groups with the goal of their maximum compression. Each iteration could be divided into two main parts: generation of possible combinations of element placements and search for the optimal placement in the set of possible combinations of placements.

Generation of possible combinations of placements is the problem from the field of combinatorics that has a factorial complexity. The number of possible placements could be calculated by the following formula:

$$C_n^k = \frac{n!}{k!(n-k)}. \tag{4}$$

For example, if the memory has $n = 100$ columns, and the memory unit size is $k = 2$, then the number of possible placements is 4950. If $k = 4$, the number of possible placements is 3,921,225. If $k = 6$, the number of possible placements is 1,192,052,400. One may conclude that this problem could not be solved by the complete enumeration method. Therefore, to achieve the goals laid above, the authors decided to use the approximation algorithm.

Searching for the possible combinations of optimal placements in the set is a partial case of the travelling salesman problem. This problem belongs to the class of NP-hard ones and has a factorial complexity. It could be solved by the branch-and-bound method or similar, however, the authors decided not to use the above methods giving preference to the greedy algorithms due to their simplicity and high speed.

Algorithm accuracy and speed are controlled by the values *max_c* and *min_c*. The requirements to them are as follows: *min_c* must not be larger than *max_c*, but not less than C_R^2. Otherwise, this algorithm will have no convergence, and, hence, will last forever. If one at each step puts *max_c* and *min_c* equal to the number of combinations C_R^2 then we obtain the fastest variant of this algorithm.

5 Data Grouping Algorithm Complexity Assessment

Algorithm complexity is one of the most principal characteristics of any algorithm. Consider the data grouping algorithm complexity in the case when the values *max_c* and *min_c* are equal to C_R^2.

If at each step *max_c* and *min_c* are equal to C_R^2, then the number of combinations at each step becomes equal to $\approx R^2/2$ from formula (1).

The greedy algorithm complexity could be assessed on the basis of its components as follows:

- search for the minimal group g. The search complexity is linear, in the worst case one needs $R/2$ operations to find the minimal group;
- provide execution of the condition $g \cap Go = \emptyset$ as follows: mark groups g that contain used elements in the set Go. To do this, it is necessary to execute additionally $R/2$ operations to set an appropriate flag after finding the minimal element;
- if the values *max_c* and *min_c* are equal to C_R^2, the value s will be 2, hence, we have to find $R/2$ minimal groups.

Thus, the total number of operations for the greedy search at the worst scenario could be written as:

$$\frac{R}{2}\left(\frac{R^2}{2}+\frac{R^2}{2}\right)=\frac{R^3}{2}. \tag{5}$$

For the expanded greedy algorithm, this number of operations must be multiplied by the number of elements $R^2/2$:

$$\frac{R^2}{2}\left(\frac{R^3}{2}\right)=\frac{R^5}{2}. \tag{6}$$

Let us denote the number of operations as n. Because at each next step the value R will be less twice, n will also decrease 2 times. This allows the above sequence of steps to be presented as a finite decreasing geometric progression with the number of elements equal to $\log_2 wc$, in which the first element is $b_1 = n$ whereas the determinant is $q = 1/2$:

$$n+\frac{n}{2}+\ldots+\frac{n}{2^{\log_2 wc-1}}. \tag{7}$$

Inserting the data into the geometric progression formula:

$$S = b_1/(1-q), \tag{8}$$

we determine the number of operations according to the following expression:

$$S = \frac{n}{(1 - 1/2)} = 2n. \tag{9}$$

Substitute n by the number of commands at each iteration taking into account that at each step we have to calculate additionally $R^2/2$ compression operations with the complexity C. For each of them we obtain the further complexity values.

Thus, the complexity of the data grouping algorithm with the use of the greedy algorithm could be described by the following expression:

$$O\left(2\left(\frac{R^3}{2} + \frac{R^2}{2}C\right)\right) = O(R^3 + R^2C). \tag{10}$$

The complexity of the data grouping algorithm with the use of the expanded greedy algorithm could be described by the following expression:

$$O\left(2\left(\frac{R^5}{2} + \frac{R^2}{2}C\right)\right) = O(R^5 + R^2C). \tag{11}$$

For other variations of the data grouping algorithm, when the values *max_c* and *min_c* are equal to the other ones, the complexity is calculated in the similar manner.

6 Results of Experimental Studies

In order to confirm the efficiency of the suggested algorithm, the authors have carried out the experimental studies. The idea of this experiment was as follows: the ROM models obtained as a result of generation by the high-level design tool Chameleon [5] were processed, and the optimized ROM models were synthesized in the target FPGA. The logic synthesis environment Quartus II 13.1 and FPGA Cyclone V 5CGXFC9E7F35C8 from Altera were used in this study.

The results of the experimental studies for the 143-bit capacity ROM with the 51,200 cells height using data compression by the dictionary method are given in Table 1, those using the sequence length coding method are given in Table 2, while

Table 1 Results of data compression using the dictionary method

No	Subunit size	Compression ratio with no grouping, %	Compression ratio using greedy algorithm, %	Compression ratio using expanded greedy algorithm, %
1	16	57.06	53.61	48.38
2	32	44.41	49.70	37.69
3	36	40.01	44.17	32.79

Table 2 Results of data compression using the sequence length coding method

No	Subunit size	Compression ratio with no grouping, %	Compression ratio using greedy algorithm, %	Compression ratio using expanded greedy algorithm, %
1	16	109.52	51.31	49.87
2	32	112.09	71.57	70.77
3	36	100.00	55.34	54.60

those using a combined sequence length coding and dictionary method are quoted in Table 3. Comparison of the results of the hardware implementation of suggested algorithm and the LZW [3] and Huffman [4] ones is shown in Table 4. The results of the optimized model synthesis are presented in Table 5.

Table 3 Results of data compression using the combined method

No	Subunit size	Compression ratio with no grouping, %	Compression ratio using greedy algorithm, %	Compression ratio using expanded greedy algorithm, %
1	16	57.07	32.47	32.33
2	32	43.88	38.01	33.76
3	36	40.06	42.64	28.80

Table 4 Comparison of the results of the hardware implementation of the compression algorithms

Comparison criterion	Dictionary	Sequences	Combined	LZW	Huffman
Frequency, MHz	165	150	180	300.661	100
Logical cells	N/A	N/A	99	287 + 283	13,824
Word size	144	144	144	8	16
Embedded RAM memory, KB	N/A	N/A	2620	252	34.56
Units required	1	1	1	18	9
Additional storage for compressed memory	No	No	No	Yes	Yes
Time of access to the memory cell, cycles	2	2	3	unstable	4
Method compression ratio, %	32.79	49.87	28.80	19.39	39.18
Random access	Yes	No	No	No	No
End device maximum frequency	550	550	550	700	491

Table 5 Comparison of the characteristics of the initial and optimized ROM models

Comparison criterion	Capacity, bit	Power consumption (without static consumption), W	Frequency, MHz
Original model	9,082,880	0.713 (0.173)	150
Optimized model	2,682,880	0.694 (0.160)	180

7 Conclusions

The problems of the efficient mapping of ROMs, being the ASPPM components, into the FPGA architecture have been considered. Suggested algorithm includes optimization of the ROM mapping into the FPGA architecture by compressing the data groups with the use of the methods described in Sects. 4.1–4.3.

As the results of synthesizing into the *Cyclone V 5CGXFC9E7F35C8* FPGA show, the optimized ROM model uses 2,682,880 bits of embedded memory and 99 logic elements and operates at the 180 MHz frequency. The compression ratio before the synthesis was 28.80 %, that after it—29.53 %. Such reduction of the above ratio is due to the necessity to place the memory content into the fixed-size units (embedded memory units).

The clock frequency of the embedded memory units and their organization have a decisive effect on the performance of devices synthesized in FPGA. The maximum clock frequency of the embedded units for FPGA used by us was 240 MHz, the maximum unit height was 8192 cells. This leads to the necessity to use an additional logic to combine the memory units. In the case when the memory unit could be fully placed into the embedded memory unit, the speed will be maximum, and the frequency of the optimized model will equal to that of the embedded memory units.

References

1. Welch, T.A.: A technique for high-performance data compression. Computer **6**(17), 8–19 (1984)
2. Huffman, D.A.: A Method for the construction of minimum-redundancy codes. In: Proceedings of the I.R.E., pp. 1098–1102, September 1952
3. Zhou, X., Ito, Y., Nakano, K.: An Efficient Implementation of LZW Decompression Using Block RAMs in the FPGA (Preliminary Version). Bull. Network. Comput. Syst. Softw—www.bncss.org, **5**(1), 12–19 (2016). ISSN 2186–5140
4. Acasandrei, L., Neag, M.: Fast parallel Huffman decoder For FPGA implementation. Acta Technica Napocensis: Electron Telecommun. **49**(1) (2008)
5. Chameleon—ASIC Design Automatic Generation Tool. Retrieved: May 30, 2016, from http://www.intron-innovations.com/?p=sld_chame

Simple Rule-Based Human Activity Detection with Use of Mobile Phone Sensors

Mariusz Fraś and Mikołaj Bednarz

Abstract The human activity recognition with use of new generation of smartphones equipped with various sensors became widely used technique lately. Many works concern especially one type of incidents—fall detection. This paper concerns detection of different activities. The proposed approach is based on use of simple rules that make possible to distinct considered events. The rules are defined with use of measured acceleration and phone orientation and take into account moments of event occurrences. The values of parameters used in rules were defined by experiment performed with use of implemented application on Android system. Finally, the accuracy of presented method is examined in the paper.

Keywords Mobile phone · Activity detection · Sensors · Accelerometer

1 Introduction

The new generations of mobile phones (or smartphones) are equipped with various sensors such as a gyroscope, accelerometer, GPS, proximity sensor, and many others. We live in the era when most people uses smartphones and many of them all the day. This permits to use such devices in many areas involved in human health, security, information and so on. There are many devices on the market dedicated to older people, which are based on the readings from connected sensors and are able to monitor different health parameters. That is very expensive solutions, however. Using smartphones we can do such monitoring easier and cheaper.

One of the very often investigated area is human activity recognition. Many works take into account one very important type of incidents—fall detection. But

M. Fraś (✉) · M. Bednarz
Wroclaw University of Technology, Wroclaw, Poland
e-mail: mariusz.fras@pwr.edu.pl

M. Bednarz
e-mail: mki.1990@hotmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_4

also detection of other types of activities—walking, jogging, etc. are very useful and are considered, lately.

This paper concerns detection of several selected activities: walking, jogging, sitting, jumping and falling. One of the main goal of presented approach was the simplicity of the method—in consequence the simplicity of implementation. This permits to use few resources of programmed device and make it work longer. Android is the most popular mobile system in the world. The possibility of implementation of proposed methods on Android-based systems is also necessary nowadays. Simplicity of application is important feature for such a systems.

The paper is organized as follows. Section 2 introduces some basics of the considered subject. Section 3 presents proposed method for activity detection. Results of experiments performed with the use of the application implementing proposed approach are presented in Sect. 4. Finally, concluding remarks are given in Sect. 5.

2 Related Works

A lot of works relate to one special incident recognition i.e. fall detection. It is very important from medical monitoring point of view. Most of the works utilize one sensor—accelerometer—and the acceleration parameter A calculated according to formula (1).

$$A = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{1}$$

where A_x, A_y, A_z, are accelerations measured for three axis's of 3D space. The early works, as e.g. in [1–3], assume fall recognition when parameter A excides some threshold value. After short time of free fall phase during which parameter A draws near 0 value, the acceleration's amplitude rises to the characteristic high level, and then stabilize near the value of 1 g (acceleration of gravity). In [3] the acceleration threshold is assumed as 2.3 g. In [2] the threshold value for fall detection is determined according to user's sex and BMI factor. Apart from maximal threshold there is also considered minimal threshold. According to sex and BMI thresholds are slightly modified and detection is performed against to assumed time window. The approach permitted increase the accuracy of fall detection (called sensitivity and defined as the number of detected falls to the number of all falls ratio) from 86.75 to 92.75 %. Numerous expansions concern data acquisition and preprocessing, activity diversifying and use of additional factors or rules to recognition process. In [4] the filtering of data with median filter. In [5, 6] phone orientation is also taken under consideration. In [5] the acceleration at the absolute vertical direction is calculated with use of A_x, A_y, A_z and azimuth, pitch and roll angles. Additionally forward lateral and backward falls are distinguished, and placement the phone on chest, waist and thigh is considered. The percentage of not detected falls was about from 2 to 10 % and the percentage of detected other activities as

falls was about from 9 to 11 %. [6] considers four categories of falls depending of what user currently does: hold by hand while typing SMS, hold by hand while normal talk, wearing in chest pocket, wearing in paints pocket. The total accuracy for forward backward and aside falls was 85 %.

The works that takes into account other activities are more complex and demand advanced data analysis. The work [7] defines eight characteristic features of measured uses signal and neural network to for activity pattern classifier. In the work [8] a lot of data is collected first, and a number of parameters as e.g. standard deviation, binned distribution, time between peaks, etc. are calculated for 10-second intervals to induce a predictive models for activities. Different learning algorithms are considered. In [9] human postures are classified using principles of kinematical theory and hierarchical rule-based algorithm. All above approaches increase accuracy of activity recognition, however demand more advanced processing devices which not always are efficient enough.

3 Detection Method

3.1 The General Procedure of Activity Recognition

Our aim is to detect the following activities:

- walking,
- jogging (running),
- fall,
- jump,
- sitting down.

The "sitting down" we detect as single short time behavior—not sitting as long time behavior. We assume that sitting lasts as long as next activity (which ends the previous one) is detected. This approach can be applied to other activities too. It is convenient for the activity that have difficult to detect feature (or parameter(s)) when it lasts, but easier characteristic to detect when it starts, and it is obvious that when it finishes then detected characteristic distinctly changes. For instance when sitting down is detected then e.g. standing or running will not start without clear change of monitored parameters. As we assumed when next activity is detected (e.g. walking in our example) then we know that sitting down is finished.

The general scheme of proposed approach is the following:

- the detection process is performed in regular phases (usually two or three),
- during first phase the first characteristic occurrence for given activity is discovered—the detected feature can be shared by more than one activity,
- when the occurrence is detected the flag of first phase detection is set and second phase begins,

- during second phase the next characteristic of given activity is tested—this can make a distinction of detected activity from the set of activities determined by the first phase,
- when the next characteristic is detected the flag of the second phase is set and if necessary next phase begins,
- the number of phases depends on the discovered characteristics and activity,
- after detection of given activity all flags can be reset and detection may start again.

The detection of second and optionally next characteristics must occur in given strictly defined period of time. The intervals depend on detected characteristic in previous phase—i.e. it depends on what activity is being recognized.

All rules are defined assuming that the phone is kept in pocket in trousers and its usual position is vertical. However, most of rules will work in other cases when we change terms in rules "horizontal" and "vertical" with the term "90° change of orientation".

3.2 Rules for Activity Detection

Sitting down

The sitting down activity is quite simple to detect when we do it as the process of short time behavior detection. As shown in Fig. 1 the monitoring records only one large rapid change of acceleration.

After one single short increase of acceleration no change is detected. The orientation of the phone is vertical at t_0 and horizontal at t_1 and t_2. The highest measured value of acceleration of vigorous sitting down was 2.1 g. The rest of peaks were within 1.2÷1.5 g.

The detection rule is the following:

1. Phase1
   In $t = t_0$: $Th_{min} < A < Th_{max}$
2. Phase2
   In $t = t_1$: $t_1 - t_0 >= \Delta t_{S,1}$ and phone orientation is horizontal
3. Phase3
   In $t = t_2$: $t_2 - t_1 >= \Delta t_{S,2}$ and phone orientation is still horizontal

The phone orientation is checked twice in order to distinguish other similar activities. The values of time periods were chosen on the basis of performed experiments.

The values of thresholds Th_{min} and Th_{max} define the interval for highest peaks for all activities (one for sitting down, fall, and jump, and repetitive peaks for walking and jogging). They are unique for all activities and are presented in Sect. 4. In rules they are noted the same way, however.

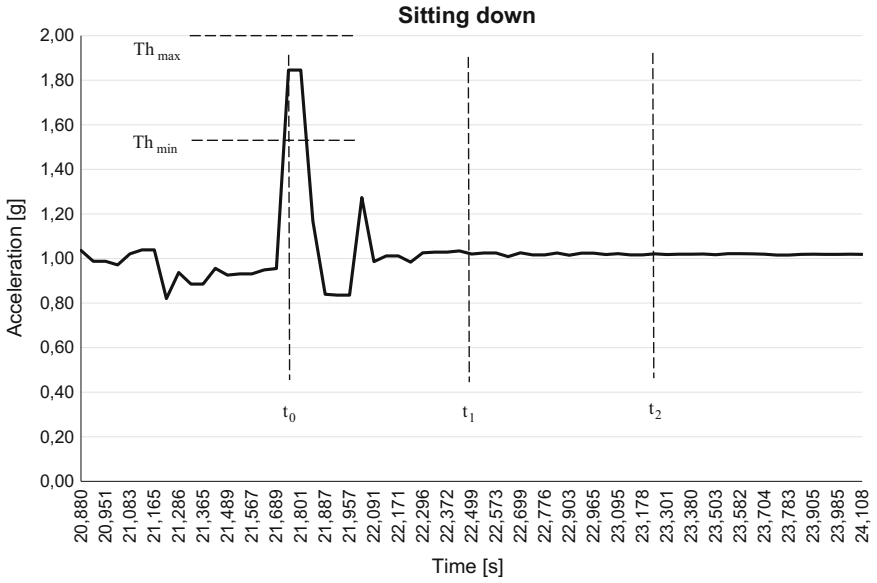


Fig. 1 Acceleration graph for sitting down activity

The time intervals in all cases (i.e. for all activities) are derived from initial experiments. For sitting down the values are: $\Delta t_{S,1} = 1s$, $\Delta t_{S,2} = 2s$.

Walking

The walking graph presented in Fig. 2 is not very regular but we can extract some repetitive schema—during step of one leg the acceleration value is about 1.5 g and then decreases to about 0.5 g and next increases to about 2 g (during step of second leg). The phone orientation remains constant (usually vertical when the phone is in the pocket). The highest measured value was below 3 g. The time between repetitive peaks for the same leg is similar. That's why we try to detect these points.

The detection rule is the following:

1. Phase1
   In $t = t_0$: $Th_{min} < A < Th_{max}$
2. Phase2
   In $t = t_1$: $Th_{min} < A < Th_{max}$ and $t_1 - t_0 >= \Delta t_{W,1}$ and phone orientation is vertical
3. Phase3
   In $t = t_2$: $Th_{min} < A < Th_{max}$ and $t_2 - t_0 >= \Delta t_{W,2}$ and phone orientation is vertical

If three consecutive fulfilling of above rules are detected then walking activity is recognized. The experimental time interval values are: $\Delta t_{W,1} = 1s$, $\Delta t_{W,2} = 2s$.

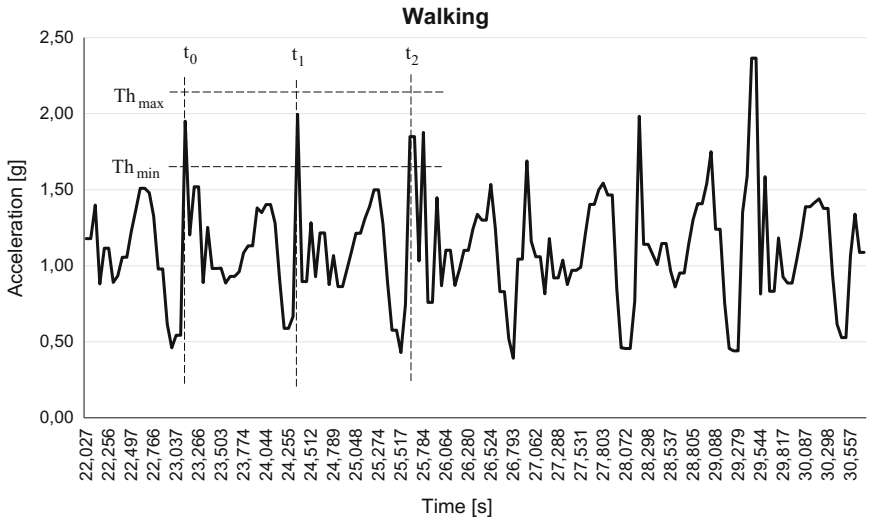


Fig. 2 Acceleration graph for walking activity

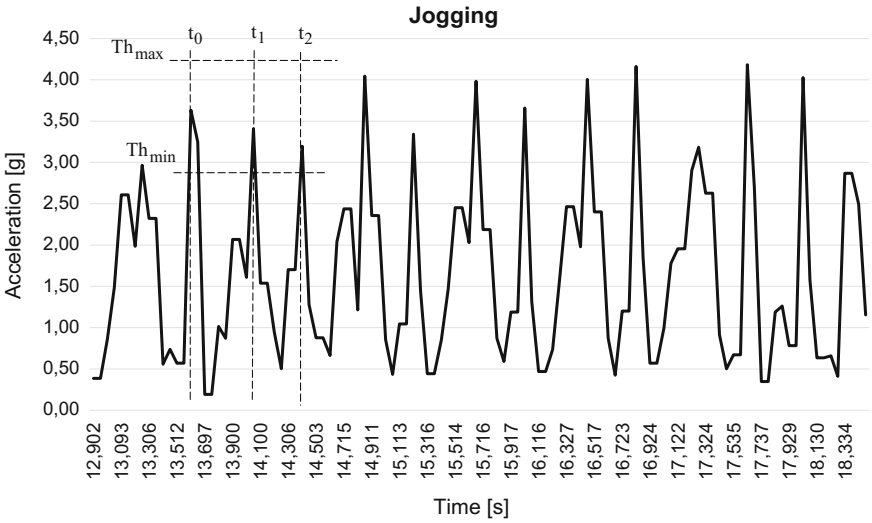


Fig. 3 Acceleration graph for jogging activity

Jogging

The Fig. 3 shows that characteristic of jogging is very similar to walking, apart from that the frequency of peaks is higher, the decrease of acceleration is when we step the other leg (in case the phone is in the pocket on the other side/leg) and the values of acceleration are distinctly higher.

The detection rule is the same as for walking except the values of parameters i.e. thresholds and time intervals $\Delta t_{R,1}$ instead $\Delta t_{W,1}$ and $\Delta t_{R,2}$ instead $\Delta t_{W,2}$ are different. The detection rule is the following:

1. Phase1
   In $t = t_0$: $Th_{min} < A < Th_{max}$
2. Phase2
   In $t = t_1$: $Th_{min} < A < Th_{max}$ and $t_1 - t_0 >= \Delta t_{R,1}$ and phone orientation is vertical
3. Phase3
   In $t = t_2$: $Th_{min} < A < Th_{max}$ and $t_2 - t_0 >= \Delta t_{R,2}$ and phone orientation is vertical

The time interval values chosen to distinct different speed of walking and jogging are the following: $\Delta t_{R,1} = 0,7\text{s}$, $\Delta t_{R,2} = 1,4\text{s}$.

Jump

The Fig. 4 shows the graph for two jumps with short interval without any activity between them. The jump consists of three phases: first—takeoff with the acceleration up to about 2÷3 g, second—when acceleration drops down to almost 0 g, and third—touchdown, when acceleration increases to 4÷5 g. The graph is a little similar to sitting down but the phone orientation is constant (usually vertical in case the phone is in the pocket) and acceleration values are distinctly higher.

Taking into account above notes, the detection rule is the following:

1. Phase1
   In $t = t_0$: $A < 0,2$ g
2. Phase2

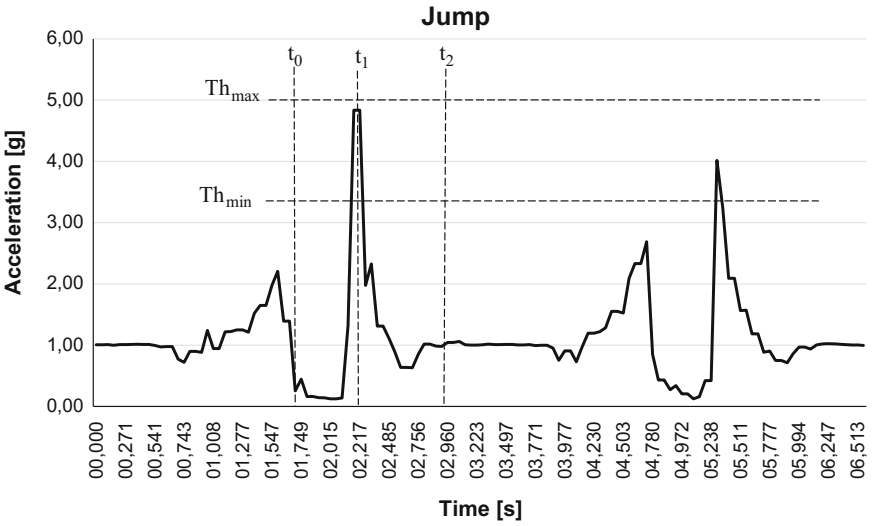


Fig. 4 Acceleration graph for jump activity

In $t = t_1$: $Th_{min} < A < Th_{max}$ and $t_1 - t_0 <= \Delta t_{J,1}$ and phone orientation is vertical

3. Phase3
   In $t = t_2$: $0.8\ g < A < 1.2\ g$ and $t_2 - t_0 >= \Delta t_{J,2}$ and phone orientation is vertical

The experimental time interval values are: $\Delta t_{J,1} = 0,7s$, $\Delta t_{J,2} = 2s$.

If three consecutive fulfilling of above rules are detected then jump activity is recognized. The condition in phase 3 for parameter *A* assumes that after jump there is no activity period and within it acceleration stabilize near 1 g. The simplified rule for jump detection consist of phase 2 and 3 only.

Fall

One of the most investigated activity is falling. The graph of acceleration (presented in Fig. 5) is similar to jump with similar values of acceleration, however there are some differences. There is no takeoff phase. The falling phase is shorter. The most important (usually utilized in most works) is that phone orientation changes to horizontal and remains horizontal for some time. The recorded peak values of acceleration were about 2.7–4.6 g. during falling the values decreased to 0.2 g.

The detection rule for the fall is the following:

1. Phase1
   In $t = t_0$: $A < 0.7$
2. Phase2
   In $t = t_1$: $Th_{min} < A < Th_{max}$ and $t_1 - t_0 <= \Delta t_{F,1}$ and phone orientation is horizontal
3. Phase3

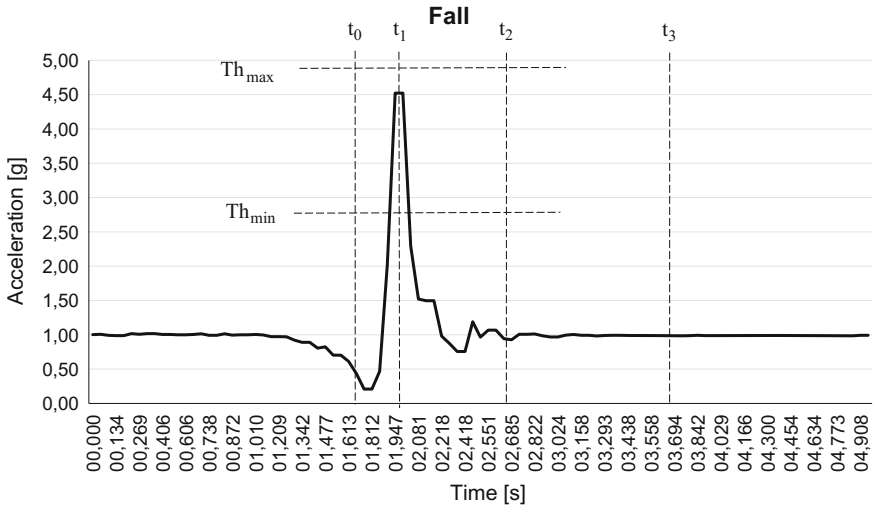


Fig. 5 Acceleration graph for fall activity

In $t = t_2$: $0.8\ g < A < 1.2\ g$ and $t_2 - t_0 >= \Delta t_{F,2}$ and phone orientation is horizontal

4. Phase4
   In $t = t_3$: $0.8\ g < A < 1.2\ g$ and $t_3 - t_0 >= \Delta t_{F,3}$ and phone orientation is horizontal

The experimental time interval values are: $\Delta t_{F,1} = 1s$, $\Delta t_{F,2} = 2s$, and $\Delta t_{F,3} = 4s$.

After the fall usually the posture remains the same for some time and the condition in phase 4 increases accuracy of fall detection. Similarly as for jump the value of parameter A stabilize near 1 g.

4 Evaluation Experiment

The evaluation experiments were performed with use of LG Nexus 4 smartphone with Android 5.0 Lollipop system. The device is equipped with different sensors among the others accelerometer, gyroscope, proximity sensor and others. The phone was kept in the pocket in pants. All experiments were performed by one 26 year old man (the comment about this fact is in final remarks).

The first series of experiments consist in testing of each activity separately to discover the range of characteristic values of acceleration and time for given activity. There was performed a dozen or so executions and the results are given in Table 1.

The threshold values in all rules were used according to values given in Table 1.

The second part of the experiment was performed to test accuracy of recognition. For every activity it was tested accuracy of recognition defined as:

$$S = \frac{TR}{TR + NR + FR} \cdot 100\% \tag{2}$$

Table 1 Characteristic values of acceleration for tested activities

Activity	Min. value of threshold Th_{min} (g)	Max. value of threshold Th_{max} (g)	Min. value of threshold (fall phase)	Max. value of threshold (fall phase) (g)
Sitting down	1.2	2.1	–	–
Walking	1.5	2.9	–	–
Jogging	2.9	4.2	–	–
Jump	3.2	4.9	0.1 g	0.15
Fall	2.7	4.6	0.2 g	0.7

Table 2 The results of accuracy evaluation experiment

	Sitting down	Walking	Jogging	Jump	Fall
TR	18	16	18	16	16
NR	2	0	0	3	0
FR	0	4	2	1	4
Accuracy	**90 %**	**80 %**	**90 %**	**80 %**	**80 %**

where: TR is correctly recognized activity, NR is not recognized activity, and FR is false recognized activity. There was performed 20 tests of every activity and TR, NR, and FR probes were counted. All activities were performed differently. For instance the "sitting down" was performed as well gently as very rapid. The results of experiment is presented in Table 2.

The overall accuracy is equal 84 %. This value is similar to many methods and slightly worse than most advanced methods.

5 Final Remarks

The proposed method of human activity detection with use of ordinary mobile phones equipped with accelerometer and gyroscope uses very simple rules, easy to implement, and consuming few device resources. It was one of the main goal of our investigations. There are three important issues worth to mention in conclusion i.e. comparison to other solutions, recognition of activities during ADL (Activities of Daily Living) and some remarks about measured characteristics.

The overall accuracy of proposed approach is slightly worse than most advanced methods. But two issues are to consider. First: the proposed approach is simple and uses much less device resources then methods performing complex analysis of data. Second: two cases: jump, and fall, were implemented in simplified form—without first phase detection. We expect that full implementation will increase accuracy.

The recognition concerning all ADLs should take into account full set of possible activities. Here we assume that when other activity is detected, it means that the previous lasts to this moment. If it would be a problem of detection some kind of activity than such monitoring will produce be false results.

Last but not least is that we noticed that some characteristic values of some activities are slightly different than in other works. This is probably caused by the fact, that sensors in different devices are differs (in the sense of sensitivity and calibration). This also suggests that threshold values should be always tuned individually. In our experiment method was tested by one man. But assuming that some adaptation of threshold values method will be applied the results of overall accuracy should be similar.

References

1. Sposaro F., Tyson G.: iFall: An android application for fall monitoring and response. In: Proceeding of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2009), Saint Paul, MN, USA, pp. 6119–6122 (2009)
2. Cao Y., Yang Y., Liu W.H.: E-FallD: A fall detection system using android-based smartphone. In: Proceeding of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012), Chongqing, China, pp. 1509–1513 (2012)
3. Tacconi C., Mellone S., Chiari L.: Smartphone-based applications for investigating falls and mobility. In: Proceeding of the 5th International Conference on Pervasive Computing Technologies for Healthcare (Pervasive Health 2011), Dublin, Ireland, pp. 258–261 (2011)
4. He Y., Li Y., Bao S.: Fall detection by built-in tri-accelerometer of smartphone. In: Proceeding of the IEEE EMBS International Conference on Biomedical and Health Informatics (BHI 2012), Hong Kong, China, pp. 184–187 (2012)
5. Dai J., Bai X., Yang Z., Shen Z., Xuan D.: PerFallD: A pervasive fall detection system using mobile phones. In: Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, Germany, pp. 292–297 (2010)
6. Viet V.Q., Lee G., Choi D.: Fall detection based on movement and smart phone technology. In: Proceedings of the IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF 2012), Ho Chi Minh City, Vietnam, pp. 1–4, (2012)
7. Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., Vecchio, A.: A smartphone-based fall detection system. Pervasive Mob. Comput. **8**, 883–899 (2012)
8. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SigKDD Explor. Newsl. **12**, 74–82 (2011)
9. Zhang, S., McCullagh, P., Zhang, J., Yu, T.: A smartphone based real-time daily activity monitoring system. Cluster Comput. **17**, 711–721 (2014)

Part II
Systems Security Issues

Timed Analysis of Security Protocols

Sabina Szymoniak, Olga Siedlecka-Lamch and Mirosław Kurkowski

Abstract This paper presents some remarks on the analysis of security protocols taking into account their time properties. Usually untimed or timed protocols are investigated with security properties, such as the secrecy of some data or the allowance of mutual authentication. These properties are independent of time. In this paper we investigate different executions, sometimes executed simultaneously, of a protocol and different types of times: ciphering and deciphering time, step execution time, whole protocol execution time, and delays in the network. Taking this into account we can investigate how these times can be chosen regardless of the possibility of an attack execution. As part of the work we have implemented a tool that helps us in the mentioned work and allows to present some experimental results.

Keywords Security protocols · Modeling and verification · Time analysis

1 Introduction

Today servers, terminals or other devices used for network communication use specially designed protocols for achieving important security goals. Every conscious network user realizes that inside such protocols safeguards are included to

S. Szymoniak (✉) · O. Siedlecka-Lamch
Institute of Computer and Information Sciences, Czestochowa University of Technology, Dabrowskiego 69/73, 42-200 Czestochowa, Poland
e-mail: sabina.szymoniak@icis.pcz.pl

O. Siedlecka-Lamch
e-mail: olga.siedlecka@icis.pcz.pl

M. Kurkowski
Institute of Computer Sciences, Cardinal Stefan Wyszynski University, Woycickiego 1/3, 01-938 Warsaw, Poland
e-mail: m.kurkowski@uksw.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_5

ensure that data transmission will be safe—data will reach the destination and will not be decoded, or taken over. On the other hand, administrators have an increasingly difficult task, because there are more and more users and data. The data often contain more and more sensitive information. The number of necessary encryption keys is growing. New protocols appear. Many parameters should be competently chosen: network protocols, security, and users so that secure communication is available within a reasonable time.

So far, the analysis of the security protocols focused mainly on one issue—whether the Intruder can carry out the attack upon some honest user or the whole network. Using different verification methods: simulation or formal modeling (inductive [11], deductive [2] and model checking [4]), it was proven whether the considered protocol is correct and resistant to the attack. There are several high-profile projects linked with model checking of security protocols such as Avispa [1], SCYTHER [3] or native VerICS [7].

However, the mentioned methods and tools usually ignore one extremely important parameter in their analysis—the time. Suppose that we have a simple protocol consisting of three steps, and it was discovered that the attack upon this protocol can be executed in ten steps. It can be concluded that the protocol is not safe. However, the protocol can be secure using a time limit calculated for three correct steps. Many protocols designers intuitively began to add timestamps and IDs to protocols. But in fact, maybe it is sufficient that the administrator possesses the knowledge: at what time or interval the protocol will be safe.

In [8] a new formal model of the protocols executions was proposed through which it is possible to test time-dependent security protocols correctness. This model is used by authors to study the authentication parameters. In Penczek and Jakubowska papers [5, 6] the network delays were taken into account. Their method was associated with the communication session proper time calculation. Tested time constraints allow the indication of the time influence on the protocol security. Mentioned studies of Penczek and Jakubowska involved only a single session and have not been continued.

In our work we develop the above considerations by analyzing the time of composing a message, the transmission time interval, the delay time in the network, and the time required to decrypt the message. All these types of time form the total time during which the protocol should be performed—the Timeout. We also analyze the attack on the protocol and how much time it will take to perform a full attack, or just a man in the middle attack.

For this study, a tool was implemented, which for a protocol given on the input, returns information about the proposed protocol execution time (so that only honest users can execute it), and about a possible attack and the time necessary to carry out the attack.

The rest of the paper is organized as follows. In the second section, we present the description of the execution's formal model, and the computational structure. In the third one, the implementation and experimental results are described and discussed. The paper ends with a summary.

2 Formal Model and Computational Structure

Time plays an important role in protocols execution. Time dependent protocols specificity shows that the protocol must be executed in a certain period of time, and time dependencies have an impact on the security. With these protocols the following concepts are related: the timestamp, the timestamp lifetime, the time of composing the message, the delay in the network and the session time.

The timestamp is a unique identifier of message generation, the value of which is set by the unit clock which sends this message. The timestamp is strongly related to its lifetime notion, which determines how long, starting from the generation moment, you can use every item in the message being sent. The time of composing the message is associated with all operations execution time, such as encryption and random number generation. The delay in the network is the time of message transmission between the sender and the recipient via a computer network, and the session time is the expected time to perform all the protocol steps.

The time conditions are related with these concepts. They define the dependencies hat relate the times of individual operations during the protocol steps execution. The inductive definition of time conditions is presented in [12]. In the same article, the protocol steps definitions were also included. We took into account the delay in the network, as well as the protocol participants' knowledge sets.

To determine the Timeout value in the individual steps as well as the maximum and the minimum session time, the following designations will be needed:

- n—the number of protocol steps,
- T_{ses}—the session time—the total time of all protocol steps,
- $T_{ses_{max}}$—the maximum session time,
- $T_{ses_{min}}$—the minimum session time,
- T_e—the time of encryption—is the time it takes for a user to encrypt the message,
- T_d—the decryption time—is the time it takes the user to decrypt the received message,
- T_g—time of generating confidential information needed to send the message,
- D—the delay in the network,
- D_{min}—minimum delay in the network—the minimum value of adopted network, delay range,
- D_{max}—maximum delay in the network—the maximum value of adopted network delay range,
- T_{out}—the timeout,
- T_{com}—the time of composing the message,
- T_k—the kth step time,
- $T_{k_{min}}$—the kth step minimum time,
- $T_{k_{max}}$—the kth step maximum time.

The message composing time is the sum of the sensitive information generation time and the message encryption time, and will be calculated from the formula:

$$T_{com} = T_e + T_g \tag{1}$$

The *k*-th step minimum time is the shortest time during which all the *k*th step components should be performed, taking into account the minimum delay in the network. Similarly, the *k*th step maximum time is the longest duration of this step, taking into account the network delay upper limit.

The *k*-th step time will obviously be calculated considering the various operations necessary to execute this step (encryption, generation, decryption), taking into account the current delay value in the network. The individual parameters values will be calculated using the following formulas:

$$T_k = T_e + T_g + D + T_d \tag{2}$$

$$T_{k_{min}} = T_e + T_g + D_{min} + T_d \tag{3}$$

$$T_{k_{max}} = T_e + T_g + D_{max} + T_d \tag{4}$$

The minimum session time is the shortest time it should take a single session for the given protocol. This parameter value is associated with the minimum network delay value and is obtained by adding up the minimum durations for all protocol steps. Similarly, the maximum session time is the longest time a single session for the given protocol should last. This parameter value depends on the maximum network delay value and is calculated by summing up the maximum durations of all protocol steps. The session time is the sum of all protocol steps times. The individual parameters values are calculated using the following formulas:

$$T_{ses} = \sum_{k=1}^{n} T_k \tag{5}$$

$$T_{ses_{min}} = \sum_{k=1}^{n} T_{k_{min}} \tag{6}$$

$$T_{ses_{max}} = \sum_{k=1}^{n} T_{k_{max}} \tag{7}$$

The Timeout value for each step should be chosen so that the honest user will have enough time to execute a step and in order to prevent the Intruder to run the attack. However, we should also remember that in the real network a disruption may appear that will have influence on the transmission time. It may also influence step correctness. The Timeout value for a single step is the sum of the maximum

values of the durations of all subsequent steps and the current step. The Timeout in a single step will be calculated according to the following formula:

$$T_{k_{out}} = \sum_{i=k}^{n} T_{i_{max}} \tag{8}$$

where:

- k—the number of the protocol step,
- i—takes values $i = k; k + 1, \ldots, n$,
- n—the number of steps,
- $T_{i_{max}}$—is the maximum time of the given step execution.

The protocol execution is deemed correct if all the time conditions imposed on the protocol steps are met, and the duration of the entire session, including the time to perform possible additional steps, will range between the minimum and maximum time of the session.

3 Implementation and Experimental Result

For experiments, an asymmetric protocol was used, which is a modification of the well-known Needham Schroeder Public Key Authentication Protocol [10], and it aims at mutual authentication of users. The scheme of this protocol in the Common Language is as follows:

$$\begin{array}{lll} \alpha_1 & A \rightarrow B: & \langle T_a \rangle K_b, i(A)\,, \\ \alpha_2 & B \rightarrow A: & \langle i(B), T_b \rangle K_a, \\ \alpha_3 & A \rightarrow B: & \langle T_b, i(A) \rangle K_b. \end{array} \tag{9}$$

In the above protocol there are two users, A and B, who want to communicate with each other. A user starts the communication by sending B its generated timestamp encrypted with the public key K_b, and its identifier i(A) is sent in the plain text.

In response, user B sent A its identifier i(B) and generated timestamp T_b, both encrypted by public key K_a. In the last protocol step, user A sends back to B the timestamp decrypted from the previous message and its ID, encrypting them by public key K_b (authenticates to B).

For the purposes of conducting the experiments, a special tool has been implemented. This tool allows the automatic modeling and generation of the security protocols executions including delays in the network. The program has been implemented in C++ using the standard library and the Standard Template Library (STL). The application includes all the assumptions described in this article, as well as in the earlier publications of the authors [7, 9, 12].

The implemented tool includes a number of classes and functions. Classes reflect all cryptographic objects, and functions enable the protocol preparation and its testing. The first step is to select the protocol for the study, then this protocol is loaded from a text file (in a ProToc format [9]), using the implemented parser. In the next step, all the possible protocols executions in the considered bounded space are generated (in different configurations, including the Intruder). For the prepared executions set, it is possible to define and use the function "ExecuteSubstitution" in order to run one of each of the generated executions.

It is also possible to limit the number of tested executions to those that represent the attack upon this protocol. The function "ExecuteSubstitution" allows to analyze the run of one of the executions, taking into account the users' knowledge, and examine such execution time. It also checks the time conditions imposed on the protocol steps.

Using the implemented tool we examined the test protocol described by expression 9. These protocol studies have been divided into three stages. In the first stage, we found an attack (the protocol is not secure). In the second stage, the encryption time impact on vulnerability to the Intruder attack was tested. In the last step, the effect of the delay on susceptibility to attack was examined. Of course, only the attacking runs have been tested.

The obtained results for this protocol are presented in Table 1. The column "Send.—Rec." contains information about the execution participants in the order in which they appear in the first protocol step. Symbols A and B indicate the honest users. I means the Intruder who appears as himself, and marks I(A) and I(B) stand for the Intruder, who pretends to be honest user. In the column "Parameters", information about the objects used by the Intruder in the execution has been posted. These objects can be random numbers, timestamps and cryptographic keys. In executions in which the Intruder is not present, the value in this column is left blank.

Table 1 Summary of the test protocol executions

No.	Send.—Rec	Parameter	No.	Send.—Rec	Parameters
1	$A \rightarrow B$		10	$B \rightarrow I(A)$	T_a, K_a
2	$B \rightarrow A$		11	$I \rightarrow A$	T_i, K_i
3	$I \rightarrow B$	T_i, K_i	12	$I \rightarrow A$	T_b, K_i
4	$I \rightarrow B$	T_a, K_i	13	$I(B) \rightarrow A$	T_i, K_b
5	$I(A) \rightarrow B$	T_i, K_a	14	$I(B) \rightarrow A$	T_b, K_b
6	$I(A) \rightarrow B$	T_a, K_a	15	$A \rightarrow I$	T_i, K_i
7	$B \rightarrow I$	T_i, K_i	16	$A \rightarrow I$	T_b, K_i
8	$B \rightarrow I$	T_a, K_i	17	$A \rightarrow I(B)$	T_i, K_b
9	$B \rightarrow I(A)$	T_a, K_a	18	$A \rightarrow I(B)$	T_b, K_b

For the second stage of research, the following assumptions for the time parameters have been made:

- the generation time assume the value of 1 time unit ([tu]),
- the network delay range is 1–3 [tu],
- the encryption and decryption time is changed for each test series from 1[tu] to 10[tu].

The time unit can be adjusted by the administrator as a given input parameter. In the case of the tested protocol, the attacking executions are those with numbers: 8, 10, 16 and 18. Executions number 10 and 18 represent the "man in the middle" attack.

Let us analyze the execution number 5:

$$\begin{array}{rrl} 8.1 & B \rightarrow I: & \langle T_b \rangle K_i, i(B), \\ 11.1 & I \rightarrow A: & \langle T_i \rangle K_a, i(I), \\ 11.2 & A \rightarrow I: & \langle i(A), T_a \rangle K_i, \\ 8.2 & I \rightarrow B: & \langle i(A), T_a \rangle K_b, \\ 8.3 & B \rightarrow I: & \langle T_a, i(I) \rangle K_i, \\ 11.3 & I \rightarrow A: & \langle T_a, i(I) \rangle K_a. \end{array} \tag{10}$$

In Table 2 an attacking execution along with its time analysis was presented. We assume the following values for specific times: $T_e = T_d = 2$ [tu], $T_g = 1$ [tu], $D = 1$ [tu]. The delay time range is 1–3 [tu].

In the 8.1 step user B sends the Intruder a message in which he put a timestamp generated by himself. For this step, we count time as a sum of: T_e, T_g, D, T_d. To execute the second step of this run (8.2) the Intruder must possess the timestamp of user A. That is why he has to execute two steps from another run (11.1 and 11.2). The time of those steps will be added to the 8.2 step time. In the 11.1 step the Intruder does not generate his timestamp because he possesses a set of prepared objects (according to the Dolev-Yao Intruder model assumptions). In response, A generates its own timestamp and sends it along with his identifier to the Intruder. The total time consists of encryption, generation, delay and decryption times.

When the Intruder possesses the needed knowledge, he can prepare cipher for 8.2 step. The time for this step consists of 11.1 and 11.2 steps times, encryption

Table 2 Times for the execution number 8

Basic step	Additional step	Time [tu]	Comment
8.1		2 + 1 + 1 + 2 = 6	ok
	11.1	2 + 0 + 1 + 0 = 5	ok
	11.2	2 + 1 + 1 + 2 = 6	ok
8.2		11 + 2 + 0 + 1 + 2 = 16	$> T_{2out}$
8.3		2 + 0 + 1 + 2 = 5	ok
	11.3	5 + 2 + 0 + 1 + 2 = 10	$> T_{3out}$ && $>T_{out}$

time, delay and decryption time. All of this together equals 16 which is more than the Timeout for 8.2 step ($T_{8:2}$out = 12). In this case, the honest user should break contact. If we consider another further possible scenario, then we can see that the Intruder has to collect the required knowledge for step 8.3, so that he can execute step 11.3. Both steps times give more than $T_{8:3}$out = 6 but also more than the maximum time of the session. The attack is not possible.

$$\begin{array}{lll} 18.1 & A \rightarrow I(B): & \langle T_a \rangle K_b, i(A), \\ 6.1 & I(A) \rightarrow B: & \langle T_a \rangle K_b, i(A), \\ 6.2 & B \rightarrow I(A): & \langle i(B), T_b \rangle K_a, \\ 18.2 & I(B) \rightarrow A: & \langle i(B), T_b \rangle K_a, \\ 18.3 & A \rightarrow I(B): & \langle T_b, i(B) \rangle K_b, \\ 6.3 & I(A) \rightarrow B: & \langle T_b, i(B) \rangle K_b. \end{array} \tag{11}$$

The second example shows "man in the middle attack". In the 18.1 step user A sends to the Intruder, impersonating on B his timestamp encoded by public key K_b together with his identifier. The total time of this step consists of generation, encoding and delay times. Because it is "man in the middle attack" the Intruder does not even try to decode message ($T_d = 0$). The Intruder needs some additional knowledge to execute the 18.2 step. He gains this knowledge in the 6.1 and 6.2 steps of another run, but he consistently does not encode, generate or decode. That is why all steps are of short duration. As was shown in table the time for every step is smaller than its timeout (Table 3).

For different times of encryption it has been shown that the Intruder can easily carry out the "man in the middle" attack. In all ten series of test executions numbers 10 and 18 were completed correctly (the time conditions in the individual steps have been met and the duration of the session was in the range of $\langle T_{ses_{min}}; T_{ses_{max}} \rangle$). Other attacking executions (8 and 16) ended incorrectly (Fig. 1).

To prevent the Intruder from carrying out the "man in the middle" attack, we should change the Timeout in the second step of the tested protocol. Performing the second step requires the Intruder to gain additional knowledge. In Fig. 2 a graph depicting the range of changes in the value of Timeout in the second step is shown.

Table 3 Times for the execution number 18

Basic step	Additional step	Time [tu]	Comment
18.1		2 + 1 + 1 + 0 = 4	ok
	6.1	0 + 0 + 1 + 2 = 3	ok
	6.2	2 + 1 + 1 + 0 = 4	ok
18.2		7 + 0 + 0 + 1 + 2 = 10	ok
18.3		2 + 0 + 1 + 0 = 3	ok
	6.3	3 + 0 + 0 + 1 + 2 = 6	ok

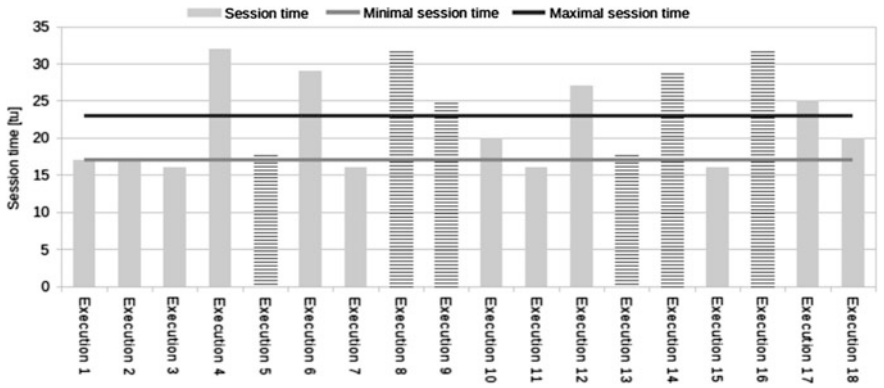


Fig. 1 Session times

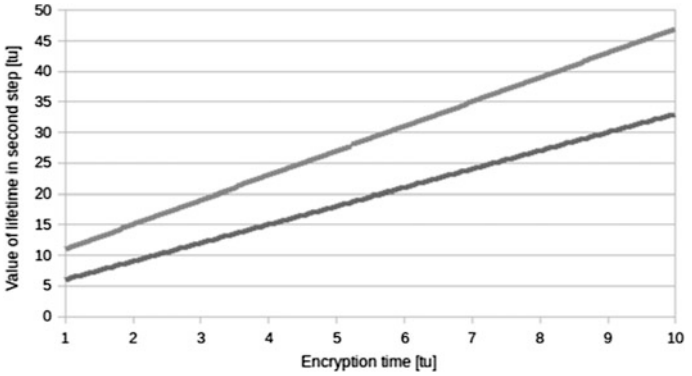


Fig. 2 Timeout changes in the second step of tested protocol

After making changes in the values of Timeout and re-testing, it was proved that the Intruder would not have sufficient time to carry out the attack.

Study of the effects of network delay on the possibility of attack was held in seven series of tests. In each series the upper limit of the delay time in the network has been changed. Of course, the lower limit of network delay has always been one unit of time—1 [tu]. In each test series the upper limit was increased by 1 [tu] from the value 4 to 10 [tu]. The fixed encryption time of 2 [tu] and the fixed value of confidential information generation time—1 [tu] was adopted.

These studies have brought interesting results: when the upper limit of the network delay ranged between 6 and 10 [tu], all attacking executions finished correctly. In turn, for delays ranges 1–4 and 1–5 the executions number 8 and 16 (attacking) were unsuccessful.

4 Summary

The analysis of the correctness of security protocols should not be restricted only to studying the capacity of a possibility to perform an attack upon it. Each protocol can be analyzed also in terms of execution time and the possibility of carrying out an attack during the time of the honest execution. In this paper we have presented a method of investigating time properties of many different executions, sometimes executed simultaneously, of the security protocols. Using this we can investigate how time parameters can be chosen regardless of the possibility of carrying out an attack execution.

We argue that *weak* protocols often can be safe if we properly select the time parameters connected with its executing. Using the presented in this paper tool networks, an administrator can analyze the quality of a considered protocol and choose the time for which it can guarantee the security of communication. It is sufficient to select the right Timeout, or establish a low level of the network delay so that the threshold for the Intruder will be difficult to jump. Looking on the other hand, knowing the delays in our own network, we can examine whether the given protocol guarantees our safety at all.

References

1. Armando, A., et. al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Proceedings of 17th International Conference on Computer Aided Verification (CAV'05), vol. 3576 of LNCS, pp. 281–285, Springer, Berlin (2005)
2. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. ACM Trans. Comput. Syst. **8**(1), 18–36 (1990)
3. Cremers, C.: The Scyther tool: verification, falsification, and analysis of security protocols. In: Proceedings of the 20th International Conference on Computer Aided Verification, Princeton, USA, pp. 414–418 (2008)
4. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–207 (1983)
5. Jakubowska, G., Penczek, W.: Modeling and checking timed authentication security protocols. In: Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'Z06), Informatik-Berichte 206(2), str. 280–291, Humboldt University (2006)
6. Jakubowska, G., Penczek, W.: Is your security protocol on time? In: Proceedings of FSEN'07, vol. 4767 of LNCS, pp. 65–80. Springer, Berlin (2007)
7. Kurkowski, M., Penczek, W.: Applying timed automata to model checking of security protocols. In: Wang, J. (ed.) Handbook of Finite State Based Models and Applications, pp. 223–254. CRC Press, Boca Raton (2012)
8. Kurkowski M.: Formalne metody weryfikacji wlasnosci protokolow zabezpieczajacych w sieciach komputerowych, wyd. Exit, Warszawa (2013)
9. Kurkowski, M., Grosser, A., Piatkowski, J., Szymoniak, S.: ProToc—an universal language for security protocols specification. Adv. Intell. Syst. Comput. **342**, 237–248 (2015)

10. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978)
11. Paulson L.: Inductive analysis of the internet protocol TLS, TR440, University of Cambridge, Computer Laboratory (1998)
12. Szymoniak, S., Kurkowski, M., Piatkowski, J.: Timed models of security protocols including delays in the network. J. Appl. Math. Comput. Mech. **14**(3), 127–139 (2015)

Some Remarks on Security Protocols Verification Tools

Mirosław Kurkowski, Adam Kozakiewicz and Olga Siedlecka-Lamch

Abstract The problem of security protocols correctness is one of the main security problems connected with communication in computer networks. Several automatic tools for verifying properties of such protocols have been proposed and used. These tools allow to find weaknesses in many variants of the protocols proposed so far. However, these tools are not entirely bug-free. In this paper, we investigate some selected problems of well known, and widely used tools for protocols verification such as AVISPA, Scyther, VerICS or PathFinder. In our considerations, we propose a few examples of protocols that cannot be used in practice or do not ensure security goals, but are positively verified by some tools. We discuss problems connected with these observations and compare how different verification tools can solve them.

Keywords Security protocols · Verification · Correctness

1 Introduction

The average network user often does not realize that under a usual exchange of views and information via the Internet complex mechanisms that protect its data are hidden. Communication protocols, and in fact their central part—the security protocols—provide communicating parties with identification (authentication), new

M. Kurkowski (✉)
Institute of Computer Sciences, Cardinal Stefan Wyszynski University,
Warsaw, Poland
e-mail: m.kurkowski@uksw.edu.pl

A. Kozakiewicz
Military Communication Institute, Warsaw, Poland
e-mail: a.kozakiewicz@wil.waw.pl

O. Siedlecka-Lamch
Institute of Computer and Information Sciences, Czestochowa University
of Technology, Czestochowa, Poland
e-mail: olga.siedlecka@icis.pcz.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_6

session keys distribution, data encryption, and data security. The user not only does not have to, but also sometimes shouldn't even know about this. It is sufficient that no one overhears, or takes over the user's data. Meanwhile, protocols invented for the needs of new tools and systems conceal construction errors. Each such protocol, before being applied, must be properly verified, and for that purpose, properly described (specified). For protocols verification, appropriate specification languages (expressing all the features of the protocol), formal models and appropriate computational structures are needed. In the last twenty years, several such tools have been introduced and used, so maybe we can consider the problem resolved. However, who can guarantee that these tools work properly?

In this paper, we examine some of today's most popular verification tools that use dedicated specification languages. Let's start with the largest of them—the AVISPA tool (Automated Validation of Internet Security Protocols and Applications) [1]. The project was carried out by several large academic centers (Genoa, Zurich, Nancy) and Siemens company. The main result of their research is a free and accessible protocol verification tool. The tool uses its own specification language HLPSL (High Level Protocol Specification Language)—a language based on roles. Protocols specification is later translated into low-level Intermediate Format (IF). The verification is performed using one of the four offered tool modules: CL-ATSE [14], OFMC [3], SAT-MC [2], TA4SP [4]. AVISPA allows to examine whether the protocol ensures information confidentiality, and authenticates users.

Another tested tool is Scyther developed by Cremers [5, 6]. The tool uses its own protocol specification language: SPDL. Scyther can be used in several ways: to verify the parameters established in the description of the security protocol, to generate and verify the security parameters of the input protocol, or for complete protocol analysis. The Scyther's author ensures that the tool correctly verifies protocols regardless of the existence of an attack. It provides an analysis of different classes of attacks, as well as the so-called multi-protocol analysis when we analyze the parallel submission of several subprotocols.

We also analyze the VerICS tool implemented by researchers at Polish scientific institutions [7, 8, 9]. The system has a module verifying security protocols, both timeless and timed protocols. The system allows testing of different execution protocol interlaces. The constructed approach examines the different executions of the same protocol as simple substitutions of different component configurations (users, keys, nonces, timestamps) to the protocol perceived as an abstract object. Formalism was designed in such a way to be able to accurately determine the correct sequences of protocol steps executions, consisting of execution interlaces. Before verification, investigated protocol should be specified in the ProToc language [8]—specially constructed for the VerICS tool. The same language is used by the PathFinder tool [12] which examines interlaces of protocol executions by using a model built on chains of states. This innovative model allows a very simple, fast and accurate analysis of security parameters, and is well suited for parallelization [13].

In the article, on the basis of two protocols, we investigate the behavior of these tools and their ability to detect incorrectness in the construction of protocols. Using this we try to stress the importance that the problem of investigation and verification

of correctness of tools constructed and devoted for automatic verification of security protocols properties is very important.

An additional goal of this paper is to show the complexity of the formal protocols description written in specially created for the presented tools specification languages. The examples show that the language HLPSL used in the AVISPA tool is very complicated and it is not intuitive. Much simpler specification language is SPDL developed for Scyther tool. The presented examples show that the ProToc language is comparable with the SPDL.

2 Automatic Verification of Correct and Incorrect Protocols

In this section we present an example of well known and correct protocol. We describe a scheme of the Needham Schroeder Public Key Authentication Protocol (NSPK), its specification in several languages and results of automatic verification done by mentioned tools.

Next we present and investigate two another protocols examples. Firstly we propose an incorrect protocol that just cannot be executed at all. A scheme of this protocol includes a bug that does not allow to executing it in real-world network environment. As the second protocol example we propose a scheme that can be executed but it does not achieve authentication goal because there exist simple attacks upon it. In both cases we show protocols schemes, their specification in several specification languages and then present results of automatic verification using the tools have done.

Example 1 Let us start by presenting a simple example: the specification and verification results of a short version of the well-known NSPK Protocol, using the aforementioned tools. This protocol consists of three steps that proceed as follows:

$$\begin{array}{lll} 1. & A \rightarrow B: & \langle N_a, i(A)\rangle K_b \\ 2. & B \rightarrow A: & \langle N_a, N_b\rangle K_a, \\ 3. & A \rightarrow B: & \langle N_b\rangle K_b. \end{array} \tag{1}$$

In the first message of the protocol, user A sends to user B a message which contains his own identifier i(A) and a newly generated pseudorandom number N_a, both encrypted under a public key of the user B. In the second message, the user B in response to A sends N_a and a newly generated by himself number N_b, both encrypted using public key K_a. In the third message, A sends a message to B which contains value N_b encrypted again using public key K_b.

It is well known that there exists an attack upon this protocol discovered by Lowe [10]. After this attack, secret data N_a and N_b are not secure, and the process of mutual authentication fails.

The specification of this protocol in SPDL language is as follows:

```
protocol nspk(A,B)
{
  role A
  {
  fresh Na: Nonce;
  var Nb: Nonce;
  send_1(A,B,{Na,A}pk(B))
;
recv_2(B,A, {Na,
Nb}pk(A));
  send_3(A,B, {Nb}pk(B));
recv_1(A,B,{Na,A}pk(B));
send_2(B,A,{Na,Nb}pk(A));
recv_3(A,B,{Nb}pk(B));
claim_B1(B,Secret,Nb);
  claim_A1(A,Secret,Na);
  claim_A2(A,Secret,Nb);
  claim_A3(A,Nisynch);
  }

role B
{
fresh Nb: Nonce;
var Na: Nonce;
claim_B2(B,Secret,Na);
claim_B3(B,Nisynch);
}
```

Observe that it is rather a simple and clear specification, easy to write by beginner users of the tool. Of course, Scyther reports the known from Lowe paper attack. No we try to verify NSPK using the AVISPA tool, therefore we need protocol's specification in the HLPSL language. This specification is much more complicated. Here we can see a fragment of the specification of one honest user: A

```
role alice (A, B: agent,
              Ka, Kb: public_key,
              SND, RCV: channel (dy))
played_by A def=
  local State : nat,
          Na, Nb: text
  init State := 0
transition
  0.  State  = 0 /\ RCV(start) =|>
      State':= 2 /\ Na' := new() /\ SND({Na'.A}_Kb)
                 /\ secret(Na',na,{A,B})
                 /\ witness(A,B,bob_alice_na,Na')
2.  State  = 2 /\ RCV({Na.Nb'}_Ka) =|>
      State':= 4 /\ SND({Nb'}_Kb)
                 /\ request(A,B,alice_bob_nb,Nb')
end role
```

In this case, all four subtools of AVISPA found the attack mentioned before.

In the ProToc language the specification is clear like in SPDL, but here we don't need to repeat information about messages sent by different users. This specification is here:

```
u=2;
p=3;
s=3;
n=1;
protocol;
A,B; i(A), nA, k+B; nA; <k+B,i(A)|nA>;
B,A; nA, nB, k+A; nB; <k+A,nA|nB>;
A,B; nB, k+B; ; <k+B,nB>;
```

Of course VerICS and PathFinder discovered Lowe's attack upon NSPK Protocol.

NSPK is an example of an incorrect protocol. Lowe proposed its corrected version in [11]. The change is in the second step, which is:

$$2. \quad B \rightarrow A : \quad \langle N_a, N_b, i(B) \rangle K_a, \tag{2}$$

Adding identifier of the user B into the second message makes this protocol correct.

Example 2 In this example we propose and investigate a protocol that cannot be executed at all by honest users. We have specified and verified this protocol with the all investigated tools. The results obtained are sometimes unexpected.

Consider a protocol named here AP1 (Aberrant Protocol 1) whose description in Common Language is as follows:

$$\begin{array}{ll} 1. & A \rightarrow B : \quad \langle N_a, i(A) \rangle K_b \\ 2. & B \rightarrow A : \quad \langle N_a, N_b \rangle K_b, \\ 3. & A \rightarrow B : \quad \langle N_b \rangle K_b. \end{array} \tag{3}$$

As in the previous example, in the first message user A sends a message to user B which contains the user's own identifier i(A) and a newly generated pseudo-random number N_a, both encrypted using public key of the user B. In the second message, B in response to A sends back N_a and N_b, both again encrypted using his own public key K_a. In the third message, A tries to send a message to B which contains value N_b encrypted again using public key K_b. Notice that the third protocol message was intended impossible to be sent. A cannot decrypt N_b after receiving the second message from B wrongly encrypted by the public key K_b.

It seems to be that a correct verification tool should notice this fact, and answer with the information that this protocol is incorrect, meaning that it cannot be executed by honest users at all.

We have checked how the aforementioned verification tools try to verify correctness of the AP1 protocol. Firstly, we have specified this protocol in SPDL

language, and we used the Scyther tool. In the SPDL specification, each user is defined as a role. The full AP1 protocol specification in SPDL is given below.

```
protocol AP1(A,B)
{
  role A
  {
  fresh na: Nonce;
  var nb: Nonce;
  send_1(A,B,{na,A}pk(B))
;
recv_2(B,A, {na,
nb}pk(B));
  send_3(A,B, {nb}pk(B));
  claim_A1(A,Secret,na);
  claim_A2(A,Secret,nb);
  claim_A3(A,Nisynch);
  }
role B
{
fresh nb: Nonce;
var na: Nonce;
recv_1(A,B,{na,A}pk(B));
send_2(B,A,{na,Nb}pk(B));
recv_3(A,B,{nb}pk(B));
claim_B1(B,Secret,Nb);
claim_B2(B,Secret,na);
claim_B3(B,Nisynch);
}
}
```

Here Scyther finds no attack and generates no attack graph. All the claims used to verify the security properties with a status (OK) are verified as secure. Scyther has no clue that the protocol cannot be executed.

We will now check how the AVISPA tool verifies the AP1 protocol. A specification of the AP1 protocol in HLPSL is too big for presentation. Here we show only specification of the role B:

```
role bob(A, B: agent,
            Kb: public_key,
            SND, RCV: channel (dy))
  played_by B def=
    local State : nat,
            Na, Nb: text
  init State := 1
  transition
    1.  State   = 1 /\ RCV({Na'.A}_Kb) =|>
        State':= 3 /\ Nb' := new() /\ SND({Na'.Nb'}_Kb)
                   /\ secret(Nb',nb,{A,B})
                   /\ witness(B,A,alice_bob_nb,Nb')
    3. State  = 3 /\ RCV({Nb}_Kb) =|>
       State':= 5 /\ request(B,A,bob_alice_na,Na)
end role
```

SATMC, one of AVISPA modules, reported an attack upon this protocol. By SATMC this attack is as follows:

```
ATTACK TRACE
1. a -> i : {na.a}_kb
2. i -> b : {na.a}_kb
3. b -> i : {na.nb}_kb
4. i -> a : {na.nb}_kb
5. a -> i : {nb}_kb
6. i -> b : {nb}_kb
```

The same attack is reported by CL-ATSE and OFMC. It is important to note that this run cannot be named as an attack. It is well known and trivial man-in-the-middle trivial-attack when the Intruder only receives and forwards messages, doesn't possess any secrets, nor does it destroy the authentication process. Consequently, we can state that these three modules discovered a run which is not an attack. It is interesting that the fourth subtool of AVISPA TA4SP reports that this protocol is safe.

Only one subtool of AVISPA OFMC reported that this protocol cannot be executed. It seems to be contradictory with the OFMC statement about an existing attack.

At the end of investigations of the AP1 protocol, we examined it with VerICS and PathFinder. Both these tools reported that this protocol is incorrect, meaning that this protocol cannot be executed.

Example 3

In the previous example we have examined a protocol that cannot be executed, but by some verification tools it is verified as correct. In this section we present a protocol that can be executed, and is claimed to be correct by almost all investigated tools (for example, the authentication process is correct). However, we are able to show two simple attacks upon this protocol. We prove that even the authentication cannot be achieved using this protocol.

Let us consider an example protocol named AP2 (Aberrant Protocol 2) whose description in Common Language is as follows:

$$\begin{array}{lll} 1. & A \rightarrow B: & \langle N_a, i(A)\rangle K_b, \\ 2. & B \rightarrow A: & \langle N_a, N_b\rangle K_b, \ll N_b > K_b, i(B) > K_a, \\ 3. & A \rightarrow B: & \langle N_b\rangle K_b. \end{array} \tag{4}$$

In the first message, a user A sends a message to B which contains a value N_a and his own identifier i.e. $i(A)$, both encrypted using public key K_b. In the second message, B responds to A with N_a and N_b encrypted again using public key K_b. In this message, user B also sends to user A the value N_b encrypted by K_b, and then encrypted both with identity B by public key K_a. In the third message, A sends a message to B which contains value N_b encrypted again using public key K_b. In this case, we can see that the structure of this protocol allows for its execution. However, in the second step A does not know B's identity because A cannot confirm N_a when A receives a message. After receiving the third message from B, wrongly encrypted by public key K_b, A cannot make an authentication.

This protocol can be executed, but authentication is not allowed. User B cannot authenticate with the user A because A doesn't see the number N_a in the second

step. User A cannot authenticate with the user B because A doesn't know the number N_b.

It is important to note that there exists a simple attack upon this protocol discovered by VerICS and PathFinder. This attack proceeds as follows:

$$\begin{array}{lll} \text{A1.} & A \rightarrow \mathrm{I}(B): & \langle N_a,\ i(A)\rangle K_b, \\ \text{B1.} & I(A) \rightarrow B: & \langle N_a,\ i(A)\rangle K_b, \\ \text{A2.} & B \rightarrow \mathrm{I}(A): & \langle N_a,\ N_b\rangle K_b, \ll N_b > K_b, i(B) > K_a, \\ \text{B2.} & I(B) \rightarrow A: & \langle N_i 1,\ N_i 2\rangle K_b, \ll N_i 1 > K_b, i(B) > K_a \\ \text{B3.} & A \rightarrow \mathrm{I}(B): & \langle N_i 1\rangle K_b. \end{array} \tag{5}$$

Here we have an another attack where user B is not needed.

$$\begin{array}{lll} \text{A1.} & A \rightarrow \mathrm{I}(B): & \langle N_a, i(A)\rangle K_b, \\ \text{A2.} & I(B) \rightarrow A: & \langle N_i 1, N_i 2\rangle K_b, \ll N_i 1 > K_b, i(B) > K_a, \\ \text{A3.} & A \rightarrow \mathrm{I}(B): & \langle N_i 1\rangle K_b. \end{array} \tag{6}$$

Now, like previously, we examine this protocol using all the investigated tools. In SDPL language the specification is given below.

```
protocol AP2(A,B)
{
  role A
  {
    fresh na: Nonce;
    var nb: Nonce;
    send_1(A,B,{na,A}pk(B));
    recv_2(B,A, {na, nb}pk(B), {{nb}pk(B),B}pk(A));
    send_3(A,B, {{nb}pk(B)}pk(A));
    claim_A1(A,Secret,na);
    claim_A2(A,Secret,nb);
    claim_A3(A,Nisynch);
  }
  role B
  {
fresh nb: Nonce;
    var na: Nonce;
    recv_1(A,B,{na,A}pk(B));
    send_2(B,A,{na,nb}pk(B), {{nb}pk(B),B}pk(A));
    recv_3(A,B,{{nb}pk(B)}pk(A));
    claim_B1(B,Secret,nb);
    claim_B2(B,Secret,na);
  claim_B3(B,Nisynch);
  }}
```

Scyther finds no attacks and generates no attack graph. All the claims used to verify the security properties with a status (OK) are verified as secure.

Now we try to verify this protocol using the AVISPA tool. A specification of user B in HLPSL language is as follows:

```
role bob(A, B: agent,
            Ka, Kb: public_key,
            SND, RCV: channel (dy))
  played_by B def=
  local State : nat,
          Na, Nb: text
  init State := 1
  transition
    1.  State = 1 /\ RCV({Na'.A}_Kb) =|>
State':= 3 /\ Nb' := new() /\
SND({Na'.Nb'}_Kb.{{Nb'}_Kb.B}_Ka)
                      /\ secret(Nb',nb,{A,B})
                      /\ witness(B,A,alice_bob_nb,Nb')
    3.  State = 3 /\ RCV({{Nb}_Kb}_Ka) =|>
          State':= 5 /\ request(B,A,bob_alice_na,Na)
end role
```

All of AVISPA's modules: SATMC, OFMC, TA4SP, CL-ATSE stated that the AP2 protocol is safe and there are no attacks upon it.

Summarizing this Example is important to note that only VerICS [8, 9] and PathFinder [12, 13] discovered attacks upon AP2 protocol.

3 Results

In the article, the different behaviors of several protocol verification tools were examined. We showed this by examining two incorrect protocols. In the first case, we have a protocol that cannot be executed at all (some message is impossible to be constructed by the sender). In the second one, the protocol can be executed, but it does not preserve mutual authentication, and there exists some simple attacks upon it. Experimental results show that not all the investigated tools are working properly. Some of them report that these protocols are safe and correct. Some of these tools report a wrong example of an attack. Remember that the OFMC module reported that the AP1 protocol cannot be executed at all, but there exists an attack upon them.

Table 1 shows a comparison of experimental results presented below. In this table **A** denotes an existing attack, **N**—no existing attack, **WA** denotes a wrong attack, **IP**—incorrect protocol (cannot be executed).

Table 1 Experimental results for three protocols and seven verification tools/modules

Protocol	CL-ATSE	OFMC	TA4SP	SATMC	Scyther	VerICS	PathFinder
NSPK	A	A	A	A	A	A	A
AP1	WA	WA/IP	N	WA	N	IP	IP
AP2	N	N	N	N	N	A	A

We can see that not only the problem of protocols correctness is important in the area of automatic software verification, but it is also the problem of the correctness of (protocols verification) tools. It seems to be that this problem is significant.

In the next work we will explore models of protocols executions used in verification tools mentioned in the article to answer in detail the question why these tools in different ways evaluate the correctness of the same protocols, which are sometimes known, they are not properly constructed.

References

1. Armando, A., et. al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Proceedings of 17th International Conference on Computer Aided Verification (CAV'05), vol. 3576 of LNCS, pp. 281–285. Springer, Berlin (2005)
2. Armando, A., Compagna, L.: An optimized intruder model for SAT-based model checking of security protocols. In: Armando, A., Vigan'o, L. (eds.) ENTCS, vol. 125, pp. 91–108. Elsevier Science Publishers, Amsterdam (2005)
3. Basin, D., Modersheim, S., Vigano, L.: An on-the-fly model-checker for security protocol analysis. In: Proceedings of ESORICS'03, vol. 2808 of LNCS, pp. 253–270. Springer, Berlin (2003)
4. Boichut, Y., Heam, P.-C., Kouchnarenko, O., Oehl, F.: Improvements on the Genet and Klay technique to automatically verify security protocols. In: Proceedings of AVIS'04 (2004)
5. Cremers, C.: The Scyther tool: verification, falsification, and analysis of security protocols. In: Proceedings of the 20th International Conference on Computer Aided Verification, pp. 414–418. Princeton, USA (2008)
6. Cremers, C.: Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In: Proceedings of 15th ACM Conference on Computer and Communications Security (CCS 2008), pp. 119–128. ACM (2008)
7. Dembinski, P., Janowska, A., Janowski, P., Penczek, W., Polrola, A., Szreter, M., Wozna, B., Zbrzezny, A.: VerICS: A tool for verifying timed automata and estelle specifications. In: Proceedings of the 9th International Conference TACAS'03, vol. 2619 of LNCS, pp. 278–283. Springer, Berlin (2003)
8. Kacprzak, M., Nabialek, W., Niewiadomski, A., Penczek, W., Polrola, A., Szreter, M., Zbrzezny, A.: Verics 2008—a model checker for high-level languages. Artif. Intell. Stud. **5**(28), 131–140 (2008)
9. Kurkowski, M., Penczek, W.: Verifying security protocols modelled by networks of automata. Fundamenta Informaticae **79**(3–4), 453–471 (2007)
10. Lowe, G.: An attack on the Needham-Schroeder public-key authentication protocol. Inf. Process. Lett. **56**(3), 131–133 (1995)
11. Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using fdr. In: TACAS, LNCS, pp. 147–166. Springer, Berlin (1996)

12. Siedlecka-Lamch, O., Kurkowski, M., Piech, H.: A new effective approach for modeling and verification of security protocols. In: Proceedings of 21st International Workshop on Concurrency, Specification and Programming (CS&P 2012), pp. 191–202. Humboldt University Press, Berlin (2012)
13. Siedlecka-Lamch, O., Kurkowski, M., Szymoniak, S., Piech, H.: Parallel bounded model checking of security protocols. In: Proceedings of PPAM'13, vol. 8384 of LNCS. Springer, Berlin (2014)
14. Turuani, M.: The CL-ATSE protocol analyzer. In: Proceedings of RTA'06, vol. 4098 of LNCS, pp. 277–286. Springer, Berlin (2006)

Algorithmic Complexity Vulnerability Analysis of a Stateful Firewall

Adam Czubak and Marcin Szymanek

Abstract Algorithmic complexity vulnerabilities are an opportunity for an adversary to conduct a sophisticated kind of attack i.e. on network infrastructure services. Such attacks take advantage of worst case time or space complexity of algorithms implemented on devices in their software. In this paper we address potential risks introduced by such algorithmic behavior in computer networks in particular on a stateful firewall. First we introduce the idea and theoretical background for the attack. We then describe in full detail a successfully conducted attack which takes advantage of the worst case computational complexity of $O(n^2)$ of a hash table data structure used to store active sessions. The attack at hand is initiated from a network protected by an stateful firewall router feature to a remote server causing a DoS (Denial of Service) on an industry grade router. Our experimental results using a real life network topology show that by generating undetected low bandwidth but malicious network traffic causing collisions in the firewall's hash table we cause the firewall to become unreachable or even announce a segmentation fault and reboot itself.

Keywords Computer networks · Complexity attack · DoS, Denial of service · Security · Network vulnerabilities · Computational complexity

1 Introduction

The problem of guaranteeing reliability and high availability of network services is a hot research issue. Computer network devices, such as routers or servers are at risk of attacks carried out by using automated software and that is why many

A. Czubak (✉) · M. Szymanek
Institute of Mathematics and Informatics, Opole University, ul. Oleska 48,
45-052 Opole, Poland
e-mail: adam.czubak@math.uni.opole.pl

M. Szymanek
e-mail: marcin.szymanek@math.uni.opole.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_7

studies have addressed issues of malicious software detection and prevention [1–3]. A particular big threat is posed by DoS (Denial of Service) attacks which are to slow down or even block operation of a network device [4, 5]. Our research concerns a special kind of DoS attacks, i.e. algorithmic complexity attacks.

2 Algorithmic Complexity

Algorithmic complexity or in other words the computational complexity, being one of theoretical computer science fields of interest gives us the means to calculate time and space complexities of given algorithm [6]. Most commonly we denominate the computational complexity using the *big* and *little O*, *Theta* and *Omega*. For those mentioned we understand informally that:

- $T(n)$ is $O(f(n))$ means that $f(n)$ describes the upper bound for $T(n)$
- $T(n)$ is $\Omega(f(n))$ means that $f(n)$ describes the lower bound for $T(n)$
- $T(n)$ is $\Theta(f(n))$ means that $f(n)$ describes the exact bound for $T(n)$
- $T(n)$ is $o(f(n))$ means that $f(n)$ is the upper bound for $T(n)$ but that $T(n)$ can never be equal to $f(n)$

To simplify the expression of algorithm with the *big O* notation we most commonly use the best, average and worst case behavior to describe the resource usage of a given algorithm dependent on the input data. Many of algorithms currently in use have decisively different average to worst computational cost. Exactly this characteristic makes algorithms vulnerable to the so called algorithmic complexity attacks [7].

2.1 Algorithmic Complexity Attacks

Algorithmic complexity attacks introduce a new attack vector [7–10]. Attacks as such take advantage of the worst case behavior of an algorithm by providing a malicious data input forcing computational complexity exceeding either expected running time or even device capabilities. If properly executed this inevitably leads to system overload resulting in most cases in DoS which is second largest cause of monetary loss according to a survey conducted in 2014 by PricewaterhouseCoopers [11].

A task at hand for an adversary is to determine whether and what kind of potentially vulnerable algorithms are used in the attacked system. Good examples of algorithms that have significantly different average to worst computational cost are the following: binary search tree, Cartesian tree, hash table, skip list, quick sort, bucket sort, hashing, pattern matching [7]. In some cases the process of determining potentially vulnerable algorithms may be a challenge in itself (e.g. proprietary

solutions), in others (e.g. open source applications) the exact implementation is available at hand.

In this paper we focus on a hash table data structure vulnerability exploitation of a proprietary networking device.

3 Hash Table Data Structure

A hash function [12] in essence maps the universe of space of keys to a finite set. It reduces the key space. A hash function should be fast and have good distribution properties. It is not required for a hash function used in this data structure to provide any means of security.

A hash table is a data structure which is a kind of an association table used to store objects, popular for its data access speed. Associations to the stored objects are made possible thanks to the usage of a key that identifies a given object or information. The key may be the whole object or a carefully chosen part of the object, in the end the key is just a sequence of bits. High access speed is not correlated with the table size or the element position in the table. Hash tables use hash function to generate a resulting hash value, which serves as a table index.

In most cases the hash function is not complex, to avoid additional computation cost for hashing operations. In a perfect scenario a unique hash value generated by hash function is assigned to a unique information or data stored in the table. If this condition is met, the search operation has the time complexity of O(1). On the other hand, if the condition is not met, we encounter a collision which means the hash function has assigned the same hash value to two or more data in the table. So one hash value corresponds to numerous keys. In this case the hash value points to "a bucket" and in some cases the term hash and bucket may be used interchangeably. The problem of collisions can be avoided by using a perfect hash function or minimal perfect hashing [13]. Usage of such functions would unfortunately impose higher computation time and space requirements than for example modular hashing [12], used in many applications for hash generation.

Dealing with collisions in hash tables can be addressed in various ways within a bucket: separate chaining with linked lists, head cells or other structures; open addressing; coalesced hashing, cuckoo hashing, Hopscotch hashing, robin hood hashing, 2-choice hashing and others [14]. All of those solutions impose additional computation on hash table operations. Generally speaking in case of collision the colliding data is put into the same bucket within a linked list attached to the bucket. While operation of inserting a single element in a hash table has the computational complexity is $O(1)$ (regardless of whether a collision occurred or not), if we try to search for a specific item and the bucket has n items stored within, we end up with the computational complexity of $O(n)$. It must be noted, that the search operation is invoked every time we wish to remove or update an object (Table 1).

Table 1 Hash table data structure time and space complexities evaluation [6]

Time complexity								Space complexity
Average				Worst				
Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
–	$O(1)$	$O(1)$	$O(1)$	–	$O(n)$	$O(n)$	$O(n)$	$O(n)$

In some cases we may even encounter the worst case complexity of $O(n^2)$ [12, 13].

The medial solution for hash tables complexity is to achieve an uniform or close to uniform distribution of hash values to minimize the impact of searching through computationally more complex buckets. Unfortunately such solution, even if empirically evaluated and proven by i.e. Perason's chi-squared test [15], does not provide significant improvements for hash table complexity vulnerabilities.

Load factor of a hash table is of the form $\alpha = \frac{n}{m}$ where n is the number of stored objects and m is the number of buckets utilized. A growth of the load factor indicates that the hash table becomes less efficient and eventually in some cases may lead to software failure, Denial of Service or even memory segmentation faults.

In this paper we describe an experiment in which we try to force the hash table used in a networks router to place all session objects in one single bucket, resulting in $\alpha = n$ and respectively for the computational complexity of any remove/update operations on this table to achieve $O(n)$.

3.1 Exploiting Hash Table Data Structure Computational Complexity

As mentioned, the goal of the attack on a hash table is feeding the algorithm with malicious data which would cause all data to be put into a single bucket, increasing the computational complexity of any operation from O(1) to O(n).

A commonly suggested counter-measure to protect hash tables against collisions is to randomize the hash table [10] by i.e. adding a secret value, as a parameter to the hash function. This solution makes the attack harder to perform but does not protect the algorithm [10].

Other possible countermeasures are:

1. using data structures with better pessimistic computational complexity such as: B-Tree, Red-Black Tree, Play Tree, AVL Tree [6];
2. early attack detection and prevention by dropping malicious data;
3. using cryptographically secure hash functions with additional reinforcement.

Unfortunately only the first solution removes the threat completely by accepting higher uniform O(1og n) complexity for all input cases.

4 Stateful Firewalls

A network router is a device which forwards layer 3 traffic (i.e. packets) according to its routing table rules. A stateful firewall is a device which, in addition to forwarding traffic,[1] is aware of the state of connections that it propagates. A firewall is capable of examining layer 4 (transport layer) to layer 7 (application layer) protocols and performs evaluation of the protocol conformance, blocking malicious traffic and in case of some more sophisticated firewall solutions to modifying the traversing traffic on-the-fly [16].

In order to perform stateful inspection and filtering, the firewall keeps track of connections and their states in a session table. So in essence a stateful firewall inspects all packets against a rule set, allowing or dropping these accordingly. If a packet is allowed to pass through, then the state of the connection this packet belongs to is being recorded in the object in the session table for the duration of the connection or until a predefined timeout expires. For example, the session object for a TCP connection must keep track of at least:

- Source and destination IP addresses;
- Source and destination ports;
- State of the connection;
- Connection timeout.

Each time a packet arrives it is asserted against the objects it the session table. If an according object is found, an arriving packet is identified as one that belongs to a previously initiated connection, the state and timeout fields are updated and the packet is allowed to pass through. If it is identified as initiating a connection, it triggers creation of a new session object, filling it with the above details, and adding the object to the session table.

4.1 Session Table

The session table must be implemented by a data structure which allows for rapid lookups. The performance of the firewall depends greatly on the ability to perform fast session table lookups and updates. A successful attack on this piece of firewall architecture might influence and degrade the session lookup process and other functions of this network device as well.

[1] A dedicated router usually has a wider range of features regarding tampering with routing tables and routing process in general then a dedicated firewall. On the other hand a dedicated firewall has more features regarding security then a router.

In case of industry grade routers and firewalls, for security reasons it is often not disclosed by vendors how the session table functions. In this paper we try to reverse engineer the state session table mechanics behind the Cisco® CBAC firewall feature [17].

4.2 Cisco® Context-Based Access Control

Context-Based Access Control or in short CBAC is one of features available on Cisco® routers. It allows to implement stateful firewall capabilities on routes. Generally speaking, stateful firewalls allow to monitor and track states and characteristics of network traffic passing it. In such firewall only known and active connections stored in routers memory are allowed to pass through the firewall. CBAC filters TCP and UDP packets using protocol session information gathered from the application layer. Based on information provided by Cisco® documentation [17] the CBAC as a context based firewall allows to:

- inspect traffic in one direction for network, transport, and application layer information;
- extract relevant port information;
- dynamically create access list entries for return traffic;
- close ports at the end of a connection;
- force protocol conformance for some protocols;
- implement denial-of-service (DoS) prevention mechanisms.

The commands used to get insight into the CBAC process are the following (Fig. 1).

4.3 CBAC Limitations

This mechanism has some limitations as well, for example the protocols it can inspect are limited to the following:

- Transmission Control Protocol (TCP);
- User Datagram Protocol (UDP);

```
router#debug ip inspect object-creation
router#debug ip inspect object-deletion
router#debug ip inspect events
router#debug ip inspect function-trace
```

Fig. 1 List of CBAC event monitoring and debug commands

- File Transfer Protocol;
- HTTP Protocol;
- CUSeeMe Protocol;
- H.323 Protocol (for example Microsoft NetMeeting or Intel Video Phone);
- R commands (r-exec, r-login, r-sh);
- Real Audio Protocol;
- Remote Procedure Call Protocol;
- Simple Mail Transfer Protocol;
- SQL Net Protocol;
- StreamWorks Protocol;
- TFTP Protocol;
- VDOLive Protocol.

By default the size of CBAC session state table is set to 1024, with the possibility of increasing it to 2048, 4096, or 8192.

4.4 *Exploring Cisco® CBAC Mechanism and Its Session Table*

In order to get some insight into the inners of CBAC the Cisco® IOS `debug` environment will be used (see Fig. 2). In particular the set of `debug ip inspect *` commands. In Fig. 3 follows a debug captured for creation of an allowed flow.

In the last line there is a hint, that the object for this particular connection was placed in bucket 0. The deletion of object is depicted in Fig. 4.

The `debug ip inspect object-creation` allowed us to verify which connection parameter or parameters are being used by the CBAC hash function as its keys. In order to reverse engineer the bucket association process multiple connections were generated with different source, destination IP addresses, source and destination ports. It turned out that the only variable that had influence on the bucket association process was the destination IP address. So connections targeting the same destination IP address are placed always into the same bucket, so the key used by the hash function is simply the destination IP address. At this point we did not know yet, how exactly the hash is being computed but what we know is enough to conduct a complexity attack which causes collisions in the CBAC hash table.

All that is to do is to generate a lot of connections to the same destination IP address, which will be placed in the same bucket of the hash table. Adding a single

Fig. 2 List of CBAC debug commands for object creation

```
router#debug ip inspect object-creation
INSPECT Object Creations debugging is on
router#debug ip inspect object-deletion
INSPECT Object Deletions debugging is on
router#
```

```
*Jul 19 01:51:50.455: CBAC Pak 84C65548 sis 84E9A388 ini-
tiator_addr (172.16.0.2:60252) responder_addr
(1.1.1.1:21)
initiator_alt_addr (172.16.0.2:60252) responder_alt_addr
(1.1.1.1:21)
*Jul 19 01:51:50.459: CBAC OBJ_CREATE: create sis
84E9A388
*Jul 19 01:51:50.459: CBAC OBJ-CREATE: sid 84EA694C acl
EXTERNAL_INT_ACL Prot: tcp
*Jul 19 01:51:50.459:  Src 1.1.1.1 Port [21:21]
*Jul 19 01:51:50.459:  Dst 172.16.0.2 Port [60252:60252]
*Jul 19 01:51:50.459: CBAC OBJ_CREATE: create host entry
84EA4250 addr 1.1.1.1 bucket 0
```

Fig. 3 Debug output of CBAC object creation and bucket assignment

```
*Jul 19 01:51:50.591: CBAC OBJ_DELETE: delete host entry
84EA4250 addr 1.1.1.1
router#
*Jul 19 01:52:26.071: CBAC OBJ_DELETE: delete sis
84E9A388
*Jul 19 01:52:26.071: CBAC OBJ-DELETE: sid 84EA694C on
acl EXTERNAL_INT_ACL Prot: tcp
*Jul 19 01:52:26.071:  Src 1.1.1.1 Port [21:21]
*Jul 19 01:52:26.071:  Dst 172.16.0.2 Port [60252:60252]
```

Fig. 4 Debug output of CBAC object deletion

object to the table should have O(1) complexity. So just by filling it with n objects, we will not achieve substantial delays. But later on, after the session table achieves the desired size, we will drop all the connections at once, which will impose the workload of O(n2).

5 Experimental Results

5.1 Primary Goals and Assumptions

The main idea is to perform a silent attack on a router, without ringing any bells, without any notification being sent to the system administrator via SNMP, syslog, command-line interface (CLI) or any other network management tool during the whole procedure. The attack should also leave no trace whatsoever on the router as well.

The malicious traffic flowing through the router should be recognized as perfectly legitimate traffic and it should consume negligible percent of bandwidth.

Router's configuration should be correct and without any modifications making it easier for the attacker to perform the procedure.

The attacked router should not be under heavy load from its other functions and have all its resources available to handle the malicious traffic. For this reason the topology of the environment is trivial: there are only two hosts on the network and the firewall rules are enabled on a single interface in only one direction.

5.2 *Analyzing CBAC Hash Function*

Next, out of pure curiosity, we decided to reverse engineer the hash function used by CBAC, as it is not disclosed in the literature. We expected, that the function would be a sequence of bitwise assembly operations like AND, OR XOR with a final modulo operation (%) against the default (1024) number of buckets. But it turned out we were almost completely wrong.

Again the debug ip inspect object-creation command proved invaluable for this purpose. Destination IP address to bucket associations that we observed are presented in first list in Table 2.

The first hint was the fact that the first octet of the IP address was contained within the resulting bucket number. It turned out not to be always true but it put us on the right track. To see more in detail how the inners work we decided to switch to simpler IP address scheme: the natural choice were 0.0.0.0–0.0.0.255 addresses. But these A class IP addresses are reserved and not allowed, so we turned to 1.1.1.1–1.1.1.255 subnet instead (second list in Table 2).

It is easy to see that the last bit of the IP address is somehow correlated with the last bit of the bucket ID, it seems to be cleared in the process. If we take the last bit of the third octet and perform a bitwise XOR with the last bit of the fourth octet, we

Table 2 Observed class B destination IP-to-bucket assignments; Observed class A destination IP-to-bucket assignments; Observed destination IP-to-bucket 0 assignments

Destination IP address	Bucket	Destination IP address	Bucket	Destination IP address	Bucket
172.31.0.2	177	1.1.1.1	0	170.85.170.85	0
172.31.0.3	176	1.1.1.2	3	1.2.1.2	0
172.31.0.4	183	1.1.1.3	2	2.1.1.2	0
172.31.0.5	182	1.1.1.4	5	1.255.170.85	1
172.31.0.6	181	1.1.1.5	4		
172.31.0.7	180	1.1.1.6	7		
172.31.0.8	187	1.1.1.7	6		
172.31.0.9	186	1.1.1.8	9		
172.31.0.10	185	1.1.1.9	8		
172.31.0.11	184	1.1.1.10	11		

get the last bit cleared and get the last bit of bucket ID correct. Now let's XOR the whole third octet with the fourth. In doing so we get the bucket ID.

Just to make sure we are right, let's take the $170_{10} = 10101010_2$ and $85_{10} = 01010101_2$ (the last row in the third list in Table 2). The result of (170 XOR 85) is clearly 255. And the debug confirms that the bucket 255 was selected for this flow.

So what about the first two octets? Why do these seem not to be taken into account? Let's modify the first two octets and take a look at resulting bucket IDs. Some non-arbitrary IPs were selected by intuition (the third list in Table 2).

If we take a closer look at data in the third list in Table 2, it turns out, that the first two octets are correlated by performing XOR operation as well. Now the resulting zeroes are clearly an outcome of a third XOR operation and this is the reason why these cancel each other out. So we deduced that the hash function for the CBAC firewall feature and bucket assignment must be of the form:

$$\begin{aligned} &Hash(DestIP) \\ &\quad = Hash(DestIP.Octet1, DestIP.Octet2, DestIP.Octet3, DestIP.Octet4) \\ &\quad = (DestIP.\text{Octet1XOR } DestIP.\text{Octet2}) \text{ XOR } (DestIP.\text{Octet3XOR } DestIP.\text{Octet4}) \end{aligned} \tag{1}$$

The final though on this result is, that there are only 255 buckets and not 1024 by default as documentation lets us assume. We decided to increase the size of the hash table with the following command.

Then we repeated the experiments for the IPs in Table 2. The question to answer was, how will the function discovered in (1) change under the new table size condition. It was presumed that since this command increases the size of the hash table, it would increase the number of buckets and the Table 2 would contain different IP-to-bucket associations making the function in (1) no longer valid, since the bucket amount would increase.

But it turned out this did not happen. After the hash table size was increased, the IP-to-bucket associations remained exactly the same. The hash function did not change at all. There were still only 255 buckets available. The conclusion is, that the command from Fig. 5 does not increase the amount of buckets and it is unclear what exactly its purpose is.

```
router(config)#ip inspect hashtable-size ?
  <1024-8192>  Hash table size allowed values:
<1024/2048/4096/8192>
router(config)#ip inspect hashtable-size 8192
```

Fig. 5 Increasing CBAC hash table size

5.3 Environment Design and Configuration

First it was experimentally verified, that the following procedure has no effect on the router without CBAC enabled.

In order for the procedure to yield sound results, it was decided not to use any form of simulation, emulation or virtualization whatsoever. Simulations do not reflect real-life hardware behavior. Emulating hardware would limit the procedure to a set of devices that have an emulator available. Virtualization is a very appealing solution for creating a lab environment, since it allows for rapid lab deployment and snapshot/rollback features.

Additionally recently available NFV (Network Function Virtualization) solutions supported by virtualized platforms seem to be a plausible option. Nevertheless, in real-life networks a router (or a L3 switch) is still a hardware appliance located in the data center, since only a hardware appliance offers hardware acceleration and in some cases even line-speed packet forwarding. Thus, it was decided in the end to use commonly available networking devices.

Network lab topology is shown in Fig. 6. Elements of the topology include:

1. Attacker host: Linux kali 4.3.0-kali1-amd64 #1 SMP Debian 4.3.3-5kali4 (2016-01-13) x86_64 GNU/Linux[2]
2. FTP server: Linux mint 3.19.0-32-generic #37 ~ 14.04.1-Ubuntu SMP x86_64 GNU/Linux

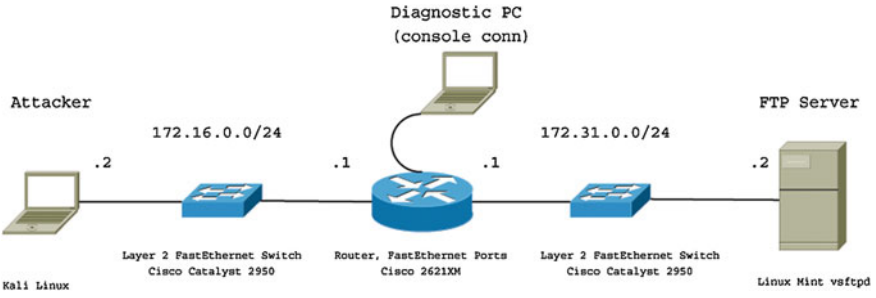


Fig. 6 Network lab topology

[2]During the creation of the lab a host with Microsoft Windows operating system was first considered as well. It turned out, that originating more than 2000 connections from the Windows host using the ftp console command is problematic and makes the system unstable. So in attempt to tackle the problem a dedicated program for the FTP client was written in MS Visual Studio. The embedded optimization methods for managing connections in .NET Framework made the program to reuse exiting connections instead of creating new ones. Shortly after this all attempts to use Microsoft Windows as an attack platform were discontinued.

3. Router: Cisco® MSR 2621XM,[3] Cisco IOS Software, C2600 Software (C2600-ADVENTERPRISEK9-M), Version 12.3(11)T)
4. Layer 2 switches: Cisco® Catalyst 2950 WS-C2950-24, Cisco IOS (tm) C2950 Software (C2950-I6K2L2Q4-M), Version 12.1(22)EA14, RELEASE SOFTWARE (fc1)

The attacking host was Kali Linux default installation with a script for the connection generation. The operating system was slightly tweaked to allow for more than the default 8192 outgoing TCP connections. The bash scripts for FTP connection generation are as follows.

In the script in Fig. 7 the "SYSTEM" FTP command was used for FTP session refresh. The command "NOOP" seams more appropriate but it is disabled in most servers, the command "PWD" generated unnecessary disk read access on the server, so the "SYSTEM" turned out to be the best, since it only sends a short string of characters to the client (Fig. 8).

In the script above a distinct delay was implemented of 0.12 s. The reason for this is to throttle down the connection generation rate to 500 per minute. It turns out that above this value the Cisco® CBAC sends a log message of the form shown in Fig. 9.

```
#!/bin/bash
if [[ $# != 3 ]] ; then
        echo "Run the program with three parameters, oth-
erwise setting default configuration -
./SingleFTPConnection 172.31.0.2 anonymous x@x.com"
  IP=172.31.0.2
  USER=anonymous
  PASSWORD=x@x.com
else
  IP=$1
  USER=$2
  PASSWORD=$3
fi
ftp -in <<EOF
open $IP
!sleep 3300
user $USER $PASSWORD
!sleep 3300
system
# !echo "Conected SingleFTPConnection PID: $$"
!sleep 3300
system
EOF
```

Fig. 7 The script SingleFTPConnection.sh for setting up a single FTP connection

[3]The Cisco 2621XM Multiservice Router was in production till 2007 and its support was discontinued as of 2013. This device was chosen deliberately, because it was never the intension of authors to show product vulnerabilities of any particular vendor. We realize that similar mechanisms are applied by other manufacturers as well. Taking a no longer offered and supported device seamed the right choice to show that the issue is of importance, while, at the same time, not causing any damage whatsoever to the manufacturers reputation.

```
#!/bin/bash
if [[ $# != 1 ]] ; then
      echo "Run the program with an amount parameter, ie.
- ./go 10 - for 10 connections"
else
  for ((i=1;i<=$1;i++))
  do
    ./SingleFTPConnection.sh &
     sleep 0.12
  done
fi
```

Fig. 8 The script GenerateConnections.sh for generating FTP connections

```
%FW-4-ALERT_ON: getting aggressive, count (2/500) current
1-min rate: 501
%FW-4-ALERT_OFF: calming down, count (0/400) current 1-
min rate: 169
```

Fig. 9 Syslog alert generated during session creation process at high rate

So to make the attack more stealth, the delay of 0.12 s was introduced between distinct connection attempts. Another reason to slow down the connection rate is the fact that the router was dropping new connections exceeding the 500 limit.

The layer 2 switches were standard off-the-shelf Cisco® Catalyst 2950 managed switches without any configuration; the default out-of-the box configuration was used so all fast Ethernet ports were contained in a single VLAN. During the attack procedure no significant drop in switching performance was noticed, the CPU usage was below 5 %, since the switching process was performed by the hardware ASICs.

The topology might be simplified if these switches were removed but those were needed to avoid the interface flapping of the FTP server, host or router. If the hosts or router resets its interface during the attack, this event is shielded from other devices thanks to the switches. If, for example, the attacking host and the router were connected directly, an interface reset on either side would have resulted in session drops or FTP client connection drops on the other side of the cable, since the operating system would have detected an interface flap and removed established connections. Another argument is that an attacker is usually not connected directly to its target, there is always a number of L2 or L3 devices along the way.

The router was configured simply to forward traffic and additionally the CBAC functionality was enabled. The inspection for the following protocols was enabled: TCP,UDP, HTTP, FTP, TFTP, ICMP. In the procedure the FTP protocol inspection was used to perform the attack (Fig. 10).

The router's state was monitored directly via a console serial connection to the minicom application on the diagnostic host. The console connection was chosen for monitoring, because any remote connections, including SSH, were dropped during the attack.

Fig. 10 CBAC configuration

```
ip inspect name EXT_INSP_RULES tcp
ip inspect name EXT_INSP_RULES http
ip inspect name EXT_INSP_RULES tftp
ip inspect name EXT_INSP_RULES icmp
ip inspect name EXT_INSP_RULES udp
ip inspect name EXT_INSP_RULES ftp
ip access-list extended EXTERNAL_INT_ACL
 deny   ip any any
interface FastEthernet0/1
ip access-group EXTERNAL_INT_ACL in
 ip inspect EXT_INSP_RULES out
```

The FTP server was a Linux Mint default installation with `vsftpd` service added. The modification were:

- Enabling access for anonymous users;
- The default value for idle_session_timeout for FTP connections was increased from 300 to 6000;
- The default value for data_connection_timeout for FTP connections was increased from 300 to 2000.
- The processor load of the FTP server never exceeded 30 %.

5.4 Procedure

The procedure was to generate a lot of connections that would be stored as objects in the CBAC hash table in a single bucket. In order to cause collisions the same destination address was used. Adding new connections is not a calculation intensive operation, as it is adding a new object at the end of the bucket, thus being harmless to the router. But after the bucket is filled with objects we close all of them at the same time. The deletion of a single object requires searching for this object in the bucket first, with presumed complexity of $O(n)$ for a single object and $O(n^2)$ for n objects. In order to do so, all of the open TCP sockets on attacker host were forced to send a TCP RST message with a simple `pkill` command. For this purpose the script in Fig. 11 was used.

It was anticipated that with the increasing number of connection objects hashed to a single bucket, forcing the router to make a lot of $O(n)$ operations would make the device become overloaded and maybe even unreachable.

Verification of the amount of sessions stored in the hash table is performed on the diagnostic PC by the command shown in Fig. 12.

Detailed attack procedure steps were the following:

1. Reboot: router, attacker PC. Restart the `vsftpd` service on the FTP server
2. Verify on the diagnostic host that the router is fully operational

```
#!/bin/bash
pkill ftp
pkill Single
pkill Generate
```

Fig. 11 The script used for generating a TCP RST flood

```
router#sh ip inspect statistics
(...)
Current session counts (estab/half-open/terminating)
[0:0:0]
(...)
```

Fig. 12 CBAC statistics output

3. Generate the required amount of connections with the script in Fig. 8
4. Verify on the diagnostic host that the router has the expected amount of sessions objects in the session table with the commands in Fig. 12
5. Start the reachability test (via ICMP ping) from attacker host to the FTP server, note the time stamps.
6. Flood the router with TCP RST with the script in Fig. 11
7. The reachability test fails at this point, wait for the reachability test to succeed and note the timestamp.
8. Continue to monitor the reachability test until the round-trip time achieves regular values of around 1.5 ms (from before the attack) for consecutive 10 s. Note the timestamp.

The time until the reachability test was successful again will be the first attribute. It denotes the amount of time in seconds it took for the router's routing function to operate again, later on denoted by unreachability time.

The time (in seconds) until the round-trip time achieves regular values is the second attribute of the attack process as it shows when the router returns to its fully functional state, later on denoted by instability time.

The procedure was performed 5 times for each of the following amount of session objects stored in a single bucket: ⟨1000,2000,3000,(…),10,000⟩.

The traffic generated by the Attacker PC was at 30 kbit/s input rate (42 packets per second) and 25 kbit/s output rate so negligible in practice.

5.5 Results

Figures 13 and 14 illustrate the time required for the router to recover after the attack. Two characteristic attributes were chosen:

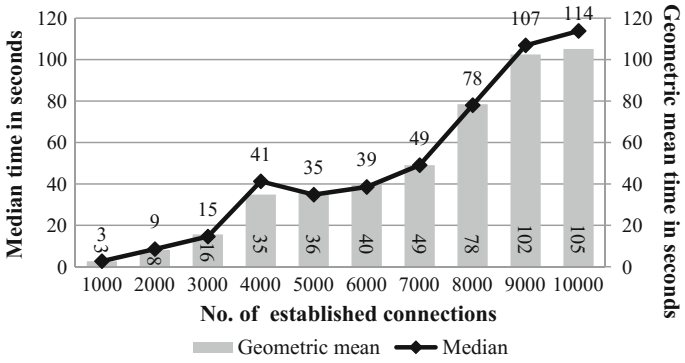


Fig. 13 Median and geometric mean times of routing service unavaliabilty

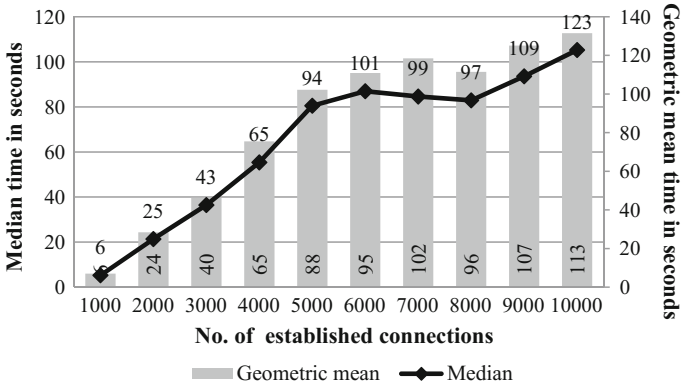


Fig. 14 Median and geometric mean times of routing service unstability

- the time it took for the router's routing function to operate again (i.e. the FTP server reachability test was successful again) (Fig. 13).
- the time it took for the routing functions to perform as expected again, i.e. the round trip time for the FTP reachability test was below 1.5 ms for consecutive 10 s (Fig. 14).

The first attribute does not convey the fact, that the network was still practically unusable. The routing function returned, but the quality of this service was unacceptable. In Fig. 15 we see this very situation. The ping should return within $\sim$1.5 ms, instead it fluctuates randomly.

It is clear, that even for a small amount of connections like 1000, forcing the router to remove those causes a 3 s of complete unavailability of the router, which is unacceptable in corporate network infrastructures.

So an attack that lasts for 120 s causes a 3 s of successful DoS. A 20 min attack results in a $\sim$2 min unavailibilty (10 % of total network unavailablity). In general,

```
[1461692031] 64 bytes from 172.31.0.2: icmp_seq=704
ttl=63 time=222 ms
[1461692031] 64 bytes from 172.31.0.2: icmp_seq=705
ttl=63 time=46.7 ms
[1461692031] 64 bytes from 172.31.0.2: icmp_seq=706
ttl=63 time=1.40 ms
(...)
[1461692034] 64 bytes from 172.31.0.2: icmp_seq=716
ttl=63 time=1018 ms
[1461692034] 64 bytes from 172.31.0.2: icmp_seq=717
ttl=63 time=827 ms
[1461692034] 64 bytes from 172.31.0.2: icmp_seq=718
ttl=63 time=1215 ms
[1461692034] 64 bytes from 172.31.0.2: icmp_seq=719
ttl=63 time=1007 ms
```

Fig. 15 Fluctuating routing operation

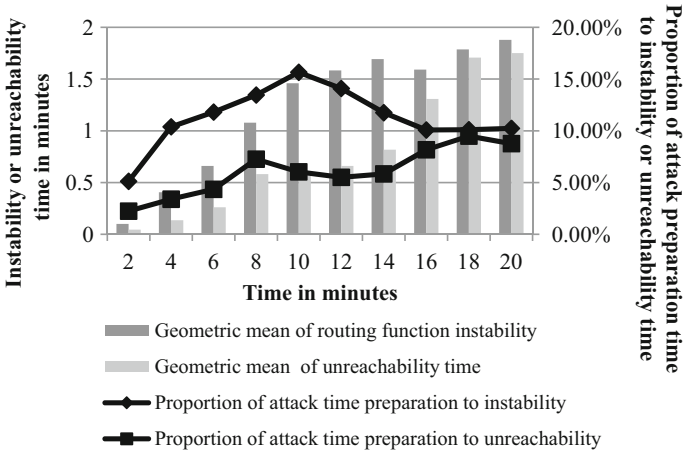


Fig. 16 DoS duration period

in Fig. 16 we see that an attack of duration above 4 and up to 10 min provides best results. Above the 10 min boundary there is no advantage in increasing the attack duration.

In Fig. 16 we can also see that a 10-min attack causes on average ~1.8 min of device instability and thus the network function unavailability.

Now we will explore the condition in which the CBAC session state table was after the attack. Figure 17 shows the amount of connection objects that the router was unable to remove from the connection state table while the attack was performed. The rapid rate of TCP RESET messages sent by the attacker's PC was too high for the router to process.

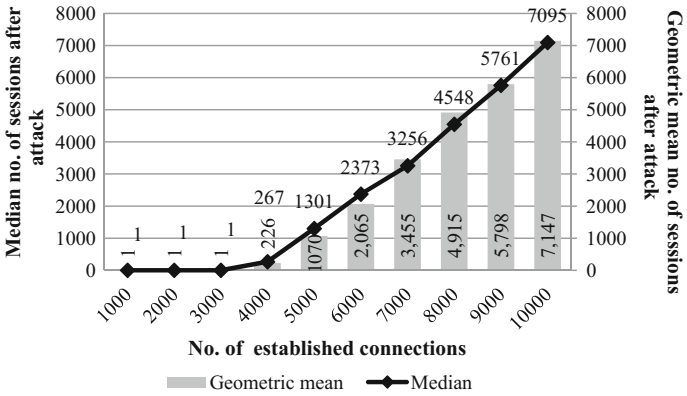


Fig. 17 Number of connections stored in memory after the attack

Figure 18 depicts the amount of successfully removed sessions by the router. It is worth mentioning that the router was capable of processing correctly no more than around 4000 sessions; above this limit session delete requests are ignored. It is not clear why; either it is an expected behavior to ignore delete requests above some safety threshold, or the router ingress queue simply cannot process that many requests and some get dropped.

As we can see, the CBAC was unable to fully recover from the attack above the 4000 connection threshold.

5.6 *Observations and Notes*

During the tests, as the amount of stored sessions reached and exceeded 7000 the TCP RST flooding caused not only the router to hang for some time, but sometimes spontaneously reboot itself with the following log output (Fig. 19).

The time it took the router to successfully reboot and return to its fully functional state was around 10 min. Unfortunately, we could not determine a pattern that would always trigger a reboot.

Further analysis revealed that simply turning the CBAC off on the router (Fig. 20), while the collisions achieved as little as 3000 session objects amount, caused the same reboot effect at every attempt (Fig. 20).

The main conclusion here is, that above 3000 objects in a single bucket boundary the CBAC mechanism is unstable and unpredictable and it is only a matter of time when it causes a segmentation fault and router reboot.

We also noticed that the longer the router was running and performing regular routing functions during our experiments with CBAC above 6000 connection threshold, the sooner a crash and reboot happened. During the experiments depicted in Figs. 14, 15, 16, 17 and 18 the router crashed just as many and as much as twice,

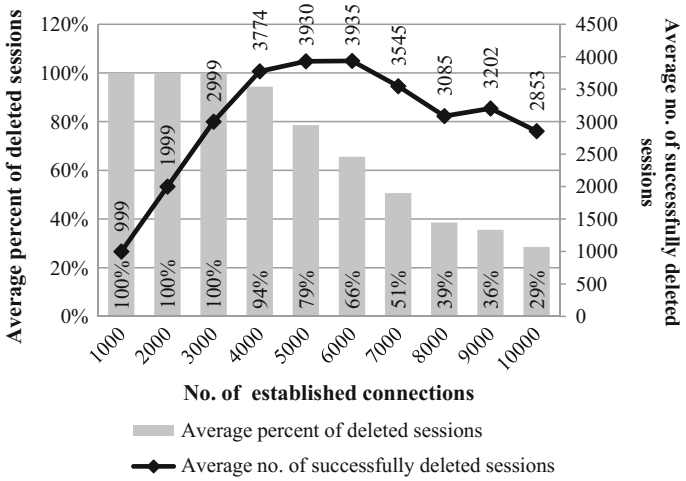


Fig. 18 Average number and percent of successfully deleted sessions

```
router(config)#
Unexpected exception to CPUvector 1200, PC = 0x821203AC,
LR = 0x821203A0
-Traceback= 821203AC
CPU Register Context:
(...)
Writing crashinfo to flash:crashinfo_20120718-225826
Unexpected exception to CPUvector 1200, PC = 0x821203AC,
LR = 0x821203A0
(...)
Nested write_crashinfo call (2 times)
=== Flushing messages (22:58:26 UTC Wed Jul 18 2012) ===
Queued messages:
*** System received a SegV exception ***
signal= 0xb, code= 0x1200, context= 0x8460a988
PC = 0x821203ac, Vector = 0x1200, SP = 0x8460bb2c
```

Fig. 19 Router reset log output

```
router(config)#
router(config)#no ip inspect
router(config)#
Unexpected exception to CPUvector 1200, PC = 0x821203AC,
LR = 0x821203A0
(…)
```

Fig. 20 Induced router reboot by CBAC shutdown

because the first step of the procedure was to reboot the equipment at startup. When the first step of the procedure was omitted, the crash and reboot happened every time the amount of objects exceeded 800.

6 Conclusions and Future Work

We have described the algorithmic complexity attacks as a possible vector which can be used for inflicting a DoS attack on computer systems. We have focused on the hash table data structure.

We have conducted and described in detail a successful DoS attack, performed on an industry-grade router using stateful firewall functionality provided by Cisco® Context-Based Access Control (CBAC). We discussed an attack scenario, all necessary scripts, commands and tools to conduct such an attack and potentially replicate our results. The attack required as little bandwidth as 24 kbit/s to perform.

Our experiments showed that it is possible to cause severe network outage of a couple of minutes or even router's crash and reboot, without leaving any trace whatsoever of the attack ever taking place. An industry grade equipment is vulnerable to algorithmic complexity attacks just as open source solutions are.

Conducted research has proven that even with corporate equipment, such as Cisco® routers, algorithmic complexity attacks are a serious threat. The behavior of an exemplary device was analyzed.

We believe that in the presented case there is a possibility to use more uniformly distributed hash function and a different key to lower the threat or impact of complexity attack without any significant increase in computational complexity of firewalls session table algorithm. We also believe that it is possible, and in some cases mandatory, to use algorithms with slightly higher computational complexity which are invulnerable to algorithmic complexity attacks without a major loss in performance. Software prone to such attacks should not be used for firewalls, IDS and IPS systems unless properly secured or used for a very narrow spectrum of controlled input.

As a follow-up we intend to develop a modified hash table data structure, customized for storing session objects representing connections in firewalls, that would be invulnerable to algorithmic complexity attacks.

References

1. Miao, R., Yu, M., Jain, N.: NIMBUS: cloud-scale attack detection and mitigation. In: Proceedings of the ACM Conference on SIGCOMM, pp. 121–122 (2014)
2. Stevanovic, D., Vlajic, N., An, A.: Unsupervised clustering of Web sessions to detect malicious and non-malicious website users. Procedia Comput. Sci. **5**, 123–131 (2011)

3. Suchacka, G., Sobków, M.: Detection of internet robots using a Bayesian approach. In: Proceedings of the 2nd IEEE International Conference on Cybernetics, Gdynia, Poland, pp. 365–370 (2015)
4. Tan, Z., Jamdagni, A., He, X., Nanda, P., Liu, R.P.: A system for denial-of-service attack detection based on multivariate correlation analysis. IEEE Trans. Parallel Distrib. Syst. **25**(2), 447–456 (2014)
5. Tao, Y., Yu, S.: DDoS attack detection at local area networks using information theoretical metrics. In: Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 233–240 (2013)
6. Thomas, H.C., Charles, E.L., Ronald, L.R., Clifford, S.: Introduction to algorithms, 3rd edn. ISBN: 9780262033848
7. Crosby, S.A., Wallach, D.S.: Denial of service via algorithmic complexity attacks. In: Proceedings of the 12th USENIX Security Symposium, pp. 29–44. USENIX Association, Berkeley, CA USA (2003)
8. Bar-Yosef, N., Wool, A.: Remote algorithmic complexity attacks against randomized hash tables. In: Filipe, J., Obaidat, M.S. (eds.) E-business and telecommunications ICETE 2007. CCIS, vol. 23, pp. 162–174. Springer, Heidelberg (2007)
9. Klink, A., Walde, J.: Efficient denial of service attacks on web application platforms (2011). https://events.ccc.de/congress/2011/Fahrplan/events/4680.en.html
10. Quynh, H.: Recommendation for applications using approved hash algorithms. NIST technical report SP 800-107. National Institute of Standards and Technology Gaithersburg, MD, US (2009)
11. US cybercrime: Rising risks, reduced readiness key findings from the 2014 US State of Cybercrime Survey, PricewaterhouseCoopers LLP (2014). http://www.pwc.com/cybersecurity
12. Sedgewick, R., Wayne, K.: Algorithms, 4th edn. Addison-Wesley Professional, Boston (2011)
13. Mehlhorn, K.: Data structures and algorithms 1: sorting and searching, vol. 1. Springer, Heidelberg (1984)
14. Babka, M.: Properties of universal hashing. Master thesis, Charles University in Prague Faculty of Mathematics and Physics (2010). http://ktiml.mff.cuni.cz/∼babka/hashing/thesis.pdf
15. Plackett, R.L.: Karl Pearson and the chi-squared test. Int. Stat. Rev. (International Statistical Institute, ISI) **51**(1), 59–72 (1983)
16. Tanenbaum, A.S., Wetherall, D.J.: Computer Networks, 5th edn. Pearson, Boston (2010)
17. Cisco IOS Security Configuration Guide: Securing the data plane. Release 12.4, Cisco Systems (2014). http://www.cisco.com/c/en/us/td/docs

Analysis of the Minutia Groups Base of Currents Algorithms 'Pasterns' Database

Michał Szczepanik, Ireneusz J. Jóźwiak, Karol Stasiński and Paweł Wichary

Abstract In this paper authors use their own fingerprint recognition algorithm in existing biometric system. Most of current systems use insecure algorithms, which can be easily broken and they not tolerate fingerprint damage. As most of new algorithms use different representation of fingerprints pattern than old one, authors present their own algorithm, which can work with existing database of fingerprints' patterns and also without changes of sensors.

Keywords Biometric · Fingerprint · Minutia group · Access controls

1 Introduction

Nowadays fingerprints recognition is one of the most popular biometric method to authorize or identify users of a system [11]. Unfortunately, existing systems based on old algorithms. As every day, people are exposed to cuts, wounds and burns; it is important that algorithms in that system are resistant to this type of damage. Existing fingerprint recognition systems usually use one of two popular algorithms' family: Minutiae Adjacency Graph (MAG) or Elastic Minutiae Matching (EMM) [2, 8]. These types of algorithms, not tolerate damages of fingerprint. Even small cut can change structure of data extracted from it [7, 12]. New algorithms which based on few types of features cannot be easily ported to existing biometric

M. Szczepanik (✉) · I.J. Jóźwiak · K. Stasiński · P. Wichary
Department of Informatics, Faculty of Computer Science and Management, Wrocław University of Technology, Wrocław, Poland
e-mail: michal.szczepanik@pwr.edu.pl

I.J. Jóźwiak
e-mail: ireneusz.jozwiak@pwr.edu.pl

K. Stasiński
e-mail: Karol.stasinski@gmail.com

P. Wichary
e-mail: wicharypawel@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_8

system, as users' data store only information about minutiae and sensors were not designed to detect them [14]. This caused that big biometric systems which existing in companies, mobile phones, or even in US visa systems need to use algorithms which are unsecured and with low level of usability. There were few concepts of updating these types of algorithms [9, 10], but each time the final solution was reregister of all users. For many big fingerprint database systems this type of solution is almost impossible. Authors' algorithm uses also only minutiae to recognize fingerprints, so based on existing data representation it should be able to convert it and recognize same fingerprints.

2 Quality Assessment of Biometric Algorithms

There are two most important performance metrics for biometric systems FAR and FRR [11, 18].

False Accept Rate (FAR), also called False Match Rate (FMR), is the probability that the system incorrectly matches the input pattern to a non-matching template from the database. It measures the percent of invalid inputs which are incorrectly accepted.

False Reject Rate (FRR), also called False Non-Match Rate (FNMR), is the probability that the system fails to detect a match between the input pattern and a matching template from the database. It measures the percent of valid inputs which are incorrectly rejected. They can be presented mathematically as:

$$\mathrm{FAR}(\mathrm{T}) = \int_{\mathrm{Th}}^{1} \mathrm{g}(\mathrm{x})\mathrm{dx} \tag{1}$$

$$\mathrm{FAR}(\mathrm{T}) = \int_{0}^{\mathrm{Th}} \mathrm{g}(\mathrm{x})\mathrm{dx} \tag{2}$$

where Th is the value of a threshold used in the algorithm and g(x) is function of algorithm that analyzed all samples with pattern. Both FAR and FRR are functions of a threshold T. When T decreases, the system has more tolerance to intraclass variations and noise, however, FAR increases. Similarly, if T is lower, the system is more secure and FRR decreases.

3 Current Algorithms

Both compared algorithms based on minutiae which are characteristic points represent ridge ending and bifurcation. Each minutia is represented by coordinates (x, y) and orientation (Θ). Orientation path is average or traces of ridges which

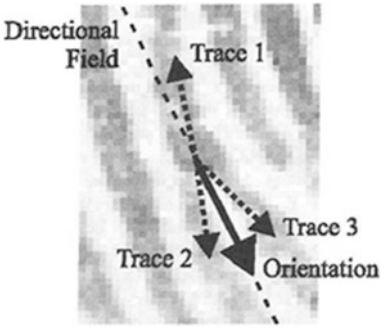


Fig. 1 Minutiae's orientation for bifurcation type

create the minutiae. Orientation of bifurcation is presented on Fig. 1. Typically, systems recognize only two basic minutiae types ridge ending and bifurcation, and ignore combinations.

3.1 Minutiae Adjacency Graph (MAG)

The most popular fingerprint recognition algorithm is based on local and global structures represented by graphs of characteristic points, see Fig. 2. In this type of algorithms, local structures to find corresponding points to align feature vector are used first, then global structures are matched [3, 4]. This type of algorithm was used by He and Ou [5], Ross et al. [15, 16]. They also use thin-plate spline (TPS) model to build an average deformation model from multiple impressions of the same finger. Owing to iteratively aligning minutiae between input and template impressions, a risk of forcing an alignment between impressions originating from two different fingers arises and leads to a higher false accept rate. Typically, a minutia matching has two steps: registration aligns fingerprints, which could be matched, as well as possible and evaluation calculates matching scores using a tolerance box between every possibly matched point (minutiae) pairs.

Database store set of vertices and set of edges, presented by Eq. 3.

$$G = (V, E), \tag{3}$$

where:

V– a set of vertices representing the set of minutiae,
E– a set of edges,
G– graph of fingerprint,
Vertex (v) represent one of minutiae according to the Eq. 4.

$$v = (x, y, \Theta) \tag{4}$$

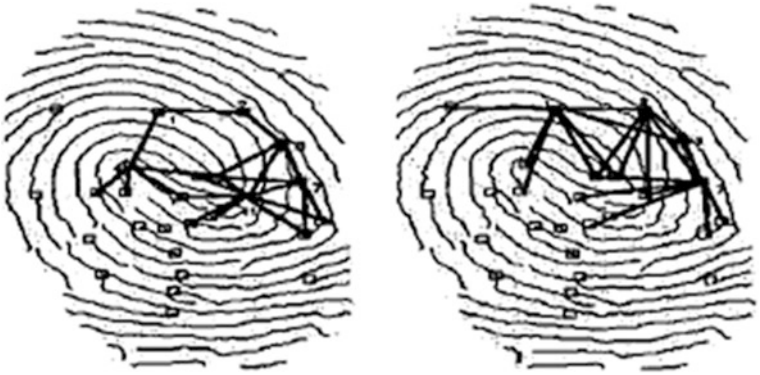


Fig. 2 Graph of minutiae used by MAG algorithm

where:

x, y– coordinates of minutiae,
Θ– minutiae's orientation.

Edges are represented by vertices which they are connecting, distance and orientation, according to the Eq. 5.

$$e = (u, v, r_{ad}, r_c, \Phi) \tag{5}$$

where:

u, v– vertices connected by edge e,
r_{ad}– Euclidean distance between vertices u and v,
r_c– number of ridges between vertices u and v,
Φ– angle between the edge and the axis x.

3.2 Elastic Minutiae Matching (EMM)

The EMM algorithm typically uses only global matching where each point (minutia) which has a type, like ending or bifurcation, needs to be matched to a related point in the second fingerprint image. Based on elastic deformations which are used to compare minutiae pairs, see Fig. 3. That are further apart because of plastic disrotations, and therefore to decrease the False Rejection Rate, so in most popular algorithms authors increase the size of bounding boxes [12, 13] to reduce this problem, but as side effect they got higher False Acceptation Rate (FAR). In this type of algorithms, for elastic match, TSP [1] also can be used, which provides better performance than only threshold of deformation. Each minutia is represented in similar way as in MAG algorithm. This is shown in Eq. 6

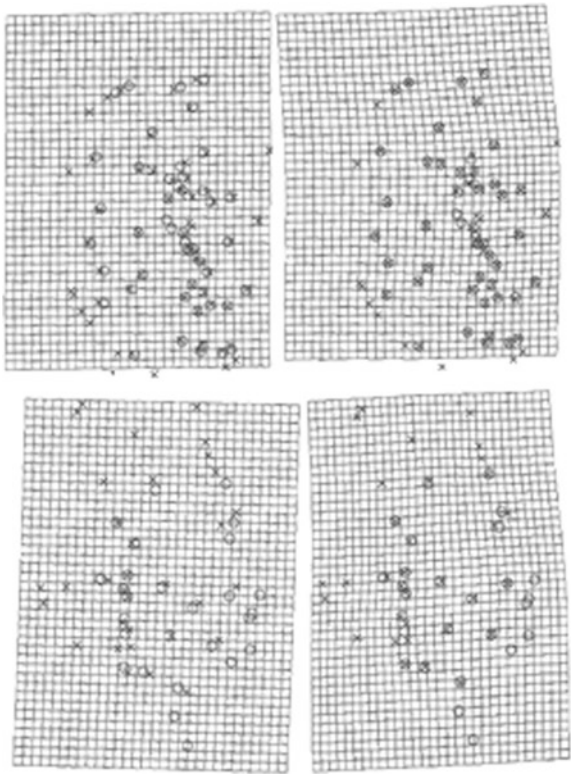


Fig. 3 Elastic Minutiae matching deformations

$$v = (x, y, \Theta) \tag{6}$$

where:

v– minutia,
x, y– coordinates of minutiae,
Θ– minutiae's orientation.

4 Fingerprint Recognition Algorithm Based on Minutia' Groups

The proposed solutions, in contrast to other algorithms, are more resistant to damage and it based on minutiae' groups instead on single minutia point, and was described in details in Author publication [17].

Fig. 4 Fingerprint divided into segments (*Source* Own work)

In this algorithm minutiae, known as Minutiae' Groups Matching, image is divided into segments. Each segment is corresponding minutiae's group and it is described by parameters (*x, y, nom*), where *x* and *y* are the coordinates of segment, and *nom* determines the number of minutiae in the group. Additionally, one of implementation uses an additional parameter specifying the probabilities of damage in a given segment, which is estimated by adjacent groups, based on the distribution of areas rejected by the mask of valid fingerprint image. Current algorithm implementation searches small groups of minutiae that that contain up to 5 minutiae (see Fig. 4). Then, based on the neighboring groups (max 4) creates a new large group. For each, the orientation parameters and the number of characteristic points are recalculated. The last step is to create a matrix of Euclidean distances between the largest groups.

Groups are compared with two parameters: dx—the distance defining the difference between groups in the pattern and tested fingerprint, px—the threshold of damage occurrence probability (determined by whether the group is under consideration in the analysis). System decide which groups should be compared and set the priority for them based on numbers of minutiae in group. After that the comparison of the groups is done. Groups are divided according to the priority, that is defined by the number of minutiae in the group and selective attention algorithms [6], which are based on probabilities of damage in a group segment. This provides quick verification of whether the analyzed fingerprint is consistent with the pattern (Fig. 5).

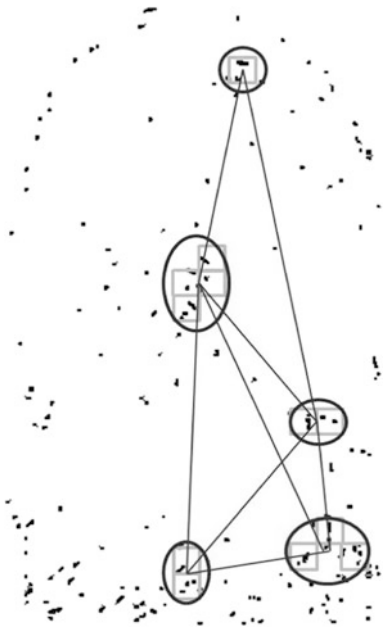


Fig. 5 Detecting relations between minutiae groups (*Source* Own work)

Developed algorithm is based on minutiae groups where each group is basically represented by the coordinates—x, y and the number of minutiae—nom contained in the group. Group covers an area equal to 2.5 the width of the furrow, its coordinates are in the middle of the square which blundering this area. Number of minutiae in the group determines its priority, additionally stored parameter describing the probability of damage—p_d in the area represented by the group. In conclusion the group is defined in Eq. 6.

$$M_g : \{x, y, nom, p_d\} \tag{7}$$

Based on these data a matrix of Euclidean distances between the groups can be created. Data on the characteristic point is limited to its weight (nom) and the probability of damage pd. Finally, Eqs. 7 can be obtained for groups relation

$$M_{gdistIJ} : \left\{M_{gI}, M_{gJ}, dist(M_{gI}, M_{gJ})\right\} \tag{8}$$

where $dist(M_{gI}, M_{gJ})$ is Euclidean distances between the groups M_{gI} and M_{gJ}.

Data stored for analysis to prevent reproduction of the original fingerprint image. Additional storage parameters to estimate the damage Allows you to better match fingerprints in the event of damage.

5 Conversion of Fingerprints' Data

Minutiae Adjacency Graph and Elastic Minutiae Matching use representation of fingerprints minutiae, see Eqs. 4 and 6. Based on that we can easily create groups representation, which are needed by Minutiae' Groups Matching algorithm.

Fingerprints area is defined by minutiae set most of the top, bottom, left and right of the image based on coordinates. Based on coordinates data are divided to segments. Each segment includes minutiae from specific area defines by parameter s, which is size of minutiae. Decision that minutia should be included in this segment is defined by Eq. 9

$$D_{add}(v) = \begin{cases} 1 & when\, x_g < x < x_g + s\, and\, y_g < y < y_g + s \\ 0 & otherwise \end{cases} \tag{9}$$

where:

$D_{add}(v)$– decision that minutiae v should be added to group (1 add, 0 do not add),
x, y– coordinates of minutia v,
x_g, y_g– coordinates of group,
s– size of group.

Both algorithm (MAG and EMM) doesn't verify fingerprints damages during template creation, this caused that it can have additional minutiae (created by incorrectly processing damaged area of fingerprint. Based on typical distribution of minutia, anomaly is a group which has 7 or more minutiae, so these types of group are ignored and marked as damaged.

6 The Experiment

Test were done using FVC2004 [11] fingerprint databases. For each of four databases, a total of 120 fingers and 12 impressions per finger (1440 impressions) were gathered. Unfortunately, most of the publicly available databases of fingerprints do not include the problem of physical damage, so additionally small damage such as cuts and burns has been generated on each sample. In most cases artificially applied damages cover 5–20 % of the fingerprint. For 10 % of the samples they cover approximately 50 % of the area to simulate severe damage. First experiment, in Table 1, shows errors probability for native implementation of algorithms. Each of

Table 1 The result of experiment using FVC2004 database with originals representations

Algorithm	FRR (%)	FAR (%)
MAG	0.80	0.63
EMM	1.20	1.15
MGM	1.70	0.86
MGM SA	0.28	0.16

Table 2 The result of experiment using FVC2004 database with representation created from MAG algorithm database

Algorithm	FRR (%)	FAR (%)
MAG	0.80	0.63
EMM	1.52	1.15
MGM	1.75	0.92
MGM SA	0.35	0.55

Table 3 The result of experiment using FVC2004 database with representation created from EMM algorithm database

Algorithm	FRR (%)	FAR (%)
MAG	0.95	0.63
EMM	1.25	1.15
MGM	1.75	0.86
MGM SA	0.31	0.16

them create their own representation of fingerprint pattern. The best values provide authors algorithm and it is three times better than MAG.

In second experiment MAG data representation was used for other algorithms, see Table 2. For each algorithms values of errors was worse than in first test, but the order of algorithms' quality stays the same.

Table 3 presents result of experiment for EMM representation of fingerprint data. For each algorithm value of False Rejection Rate grow, but only 0.05 %. The data representation of EMM stores more minutiae than MAG. In second algorithms only minutiae that create graph, so they are near to each other, are stored.

7 Conclusion and Future Work

The proposed algorithm enables the identification of fingerprints in places where they are exposed to frequent damage, without changing whole infrastructure like sensors. Proposed algorithm can be easily adopted in places where Minutiae Adjacency Graph (MAG) or Elastic Minutiae Matching (EMM) algorithms are used. The way how proposed solution stores fingerprint data makes it impossible to recreate the original structure as it only provides information about the minutiae clusters and their frequency. As future work design of algorithm, which detect anomalies in fingerprints data patterns, is planned.

References

1. Bazen, A.M., Gerez, S.H.: Fingerprint matching by thin plate spline modelling of elastic deformations. Pattern Recogn. (2003)
2. Cappelli, R., Lumini, A., Maio, D., Maltoni, D.: Fingerprint classification by directional image partitioning. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5), 402–421 (1999)

3. Chikkerur, S., Govindaraju, V., Cartwright, E.: NK-plet and coupled bfs: a graph based fingerprint representation and matching algorithm. LNCS pp. 309–315 (2006)
4. Grzeszyk, C.: Forensic fingerprint examination marks (in Polish). Wydawnictwo Centrum Szkolenia Policji, Legionowo (1992)
5. He, Y., Ou, Z.: Fingerprint matching algorithm based on local minutiae adjacency graph. J. Harbin Inst. Technol. **10**(05), 95–103 (2005)
6. Huk, M., Szczepanik, M.: Multiple classifier error probability for multi-class problems. In: Eksploatacja i Niezawodnosc—Maintenance and Reliability, vol. 3, pp. 12–17 (2011). doi:10.17531/ein.2011
7. Hicklin, A., Watson, C., Ulery, B.: How many people have fingerprints that are hard to match, NIST Interagency Report 7271? (2005)
8. Hong, L., Wan, Y., Jain, A.K.: Fingerprint image enhancement: algorithm and performance evaluation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 777–789 (1998)
9. Indovina, M., Uludag, U., Snelick, R., Mink, A., Jain, A.: Multimodal Biometric Authentication Methods: A COTS Approach. In: Proceedings of MMUA (2003)
10. Jain, A.K., Ross, A., Nandakumar, K.: Introducing to biometrics, Springer, Berlin (2011)
11. Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of Fingerprint Recognition, 2nd edn. Springer, Berlin (2009)
12. Pankanti, S., Prabhakar, S., Jain, A.K.: On the individuality of fingerprints. In: Proceedings of Computer Vision and Pattern Recognition (CVPR) (2001)
13. Parziale, G., Niel, A.A.: Fingerprint matching using minutiae triangulation. In: Proceedings of ICBA (2004)
14. Ratha, N.K., Govindaraju, V.: Advances in Biometrics: Sensors, Algorithms and Systems. Springer, Berlin (2007)
15. Ross, A., Dass, S.C., Jain, A.K.: A deformable model for fingerprint matching. Pattern Recogn. **38**(1), 95–103 (2005)
16. Ross, A., Nandakumar K., Jain, A.K.: Handbook of Multibiometrics. International Series on Biometrics. Springer, Berlin (2011)
17. Szczepanik, M., Szewczyk, R.: Fingerprint identification algorithm (in Polish). KNS **1**, 131–136 (2008)
18. Wayman, J.L., Jain, A.K., Maltoni, D., Maio, D.: Biometric Systems, Technology, Design and Performance Evaluation, 1st edn. Springer, Berlin (2005)

Part III
Computing and Service Systems Architectures

Self-organizing Agents for Dynamic Network- and QoS-Aware Service Composition in Cloud Computing

Leila Helali and Zaki Brahmi

Abstract Cloud computing plays a vital role in distributed systems used by Internet users. It provides a flexible environment in which data, equipment and services can be shared among end users in order to save time and cost. Cloud service composition is still one of the most important issues related to this paradigm. Indeed, Dynamic Cloud Service Composition (DCSC) is the process of combining a chain of connected atomic Cloud services together in order to create a more complex and value-added composite service. In this work, we present a new method of cloud service composition guided by QoS of services (execution time and cost) and network QoS (data transfer cost and latency). The latency is estimated by the Euclidean distance calculated based on the coordinates of Cloud services that are based on a network coordinate system called GNP (Global Network Positioning). The proposed solution is based on the paradigm of agents where autonomous entities cooperate together to generate an optimal composition within a reasonable time. Experimental results confirm the modesty of our approach.

Keywords Cloud · GNP · Composition · Qos · Agent

L. Helali (✉)
Higher Institute of Applied Science and Technology of Sousse, Sousse University, Sousse, Tunisia
e-mail: leilahelali.ing@gmail.com

Z. Brahmi
RIADI Laboratory, Manouba University, Manouba, Tunisia
e-mail: zakibrahmi@yahoo.fr

Z. Brahmi
Higher Institute of Computer Sciences and Communications Techniques of Hammam, Sousse University, Sousse, Tunisia

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_9

1 Introduction

Cloud computing has emerged as a new paradigm for the next generation of distributed computing. It has attracted much attention in academia and industry. The vision of cloud computing is that computing will not be conducted on local computers in the future, but in distributed installations operated by third parties IT services. Several companies like Amazon [1], Microsoft [2] and IBM [3] have already proposed cloud computing solutions on the market [4]. Indeed, cloud computing is to uniformly expose the hardware and software as a service that can be rented on demand. In this new paradigm, organizations no longer need large capital expenditure or equipment to deploy their services or software.

Thus, all the cloud resources are accessed as services, which can be classified into application services (SaaS) and utility computing services (PaaS and IaaS). Web services is a SaaS, which represent software components encapsulating units of well-defined business functionalities. Utility computing services are software or virtualized hardware that support the application services, as virtual machines, network services, etc. Indeed, in many cases self-Cloud service represents a partial solution which is not enough to meet the applicant's requirements. Thus, a service composition processes is necessary for the purpose of providing a single composite service to Cloud consumers. The Dynamic Cloud Service Composition (DCSC) problem is made to select a set of interrelated services to achieve complex functionality by combining basic services. However, DCSC is subject to several issue: (i) multiple services with equivalent functionality are available, (ii) the same service is deployed in different Clouds, (iii) highly dynamic environment like Cloud computing. Thus, finding a feasible and optimal composition in terms of time and cost, services and network perspective is an NP-complete problem [5].

We propose, in this paper, a composition approach taking into account: execution time of service, network latency and energy consumption in cloud services and network. It was based on a network coordinate system "Global Network Positioning (GNP)" [6] to estimate the network latency and multi-agent systems (MAS) to generate the optimal composition. Indeed, intelligent agents cooperate together to find dynamically optimal solutions while maintaining the distributed nature and rapidly evolving cloud environments. They self-organize according to the inputs and outputs of services to form a social network of agents.

The rest of the paper is structured as follows: Sect. 2 illustrates a study of the related work as well as their potential limitations. Section 3 elaborates the basic used concepts and the social network composition Cloud service formulation problem. We focus in Sect. 4 on the details of the proposed approach. Section 5 shows the experiment results of our approach. Finally, in Sect. 6, we conclude this paper and highlight some future work.

2 Related Work

There is many approaches to treat cloud services composition problem. Some of them adopt the QoS of Cloud services as a criterion like [4]. The major drawback of this approach is that the penalty factor in the adaptation function is static [4]. However, cloud environments are highly dynamic. In [7, 8] the authors propose a cloud service composition approach taking into account the network QoS (latency and transfert rate). Both [7, 8] are centralized methods, treat only the selection process based on a predefined composition. In [9], a multi-Cloud service composition method is presented. The service composition problem is translated into a combinatorial optimization problem. Then, by using artificial intelligence planning techniques, a service composition solution is provided. However, this solution is also based on a predefined composition and do not treat the network QoS. Other methods based on multi-agent systems took place [10, 11]. They allow you to select the most appropriate services to create a composite solution. In the solution described in [11] no QoS attribute is taken into account. These two approaches are based on a predefined composition as well and do not allow to consider the network QoS. Finally, the solution proposed in [12] provides a set of service composition encapsulated in a middleware. It seems difficult to implement and adapt them to the cloud platforms already developed and functional.

To summarize, all the mentioned approaches are based on a predefined composition where this latter is reduced into a selection problem to choose the best concrete service that will run every abstract task. Most of these approaches do not take into account the network QoS that has a significant impact on the performance of the composition. Thus, most cited approaches are less answered in our case where we treated composition in a distributed environment, open and rapidly changing. In this case, a distributed dynamic solution that addresses the specifics of the environment in question will be our favor.

In this paper, we adopted a composition solution based on the MAS. Thus our approach is based on four basic principles: (i) QoS-awareness: Cost and execution time of services, (ii)Network-awareness: the number of Cloud services and their distributions across the network increasing rapidly, it increases the impact of the network on the QoS of the composition, (iii) No predefined composition is considered: most work is based on static composition techniques defined using business process. Due to the ongoing development of the service offering and capabilities, a static description of the composition is difficult to maintain. Our approach overcomes this limitation and to adjust the composition to the expectations of the user and therefore not to be constrained by a priori composition, iv) MAS based: our approach is based on autonomous and dynamic agents that cooperate together to enable a distributed decision-making and generate the optimal composition within a reasonable time.

3 Dynamic Network- and QoS-Aware Cloud Service Composition

In our solution, we consider a user request that needs to have a response as a Cloud service composition. The completion of a composition yields a certain value, and it requires varying numbers of Cloud services of different functionalities and QoS. A user request task can be completed only if all required user constraints are *fulfilled*.

In our approach, Cloud services with similar functionalities are grouped in a Service Class (SC). Each SC is managed by an agent named Class Agent. In addition, each agent is connected to a limited number of other agents yielding a social network. For the completion of her task, an agent may enlist the resources of other agents, but only those she's connected to in the network. More formally, let $S = \{S_i | 1 \leq i \leq n\}$ denote a set of n Cloud services.

Definition 1 (*Cloud* service) For our approach, a Cloud service S_i is characterized by its Inputs I_{S_i}, Outputs O_{S_i}, QoS Q_{S_i} and Clouds C_{S_i}.The Inputs represent the information (preconditions) needed to use the service. The Outputs represent the information (effects) generated from the use of the service. The Clouds represent the set of Clouds where services deployed. Formally a Cloud service S_i is defined as (1):

$$\mathrm{Si} = (I_{S_i}, O_{S_i}, Q_{S_i}, C_{S_i}). \tag{1}$$

Definition 2 (*Query*) A query $R = (\mathrm{I^R}, \mathrm{O^R})$ is a virtual web service. The Inputs $\mathrm{I^R}$ represents information that the user can provide. The Outputs $\mathrm{O^R}$ represents the results required by the user. As example $\mathrm{I^R}$ can be a document ".WORD" and a file ".PDF". As $\mathrm{O^R}$ we want to convert the input parameters format images ".GIF".

Definition 3 (*Class Agent*) A Class Agent CA_i is an agent managing a set $CWS_i \subseteq S$ of Cloud services that provide similar functionalities (Input and Output) with possibly different QoS; formally (2):

$$\mathrm{CAi} = (\mathrm{IDi}, \mathrm{CWSi}, \mathrm{I_{CA_i}}, \mathrm{O_{C_{Ai}}}) \tag{2}$$

Definition 4 (*Social network*) An agent social network $SN = (V, ACE)$ is a dependency graph; where vertices V are agents class, and each edge $(i, j) \in ACE$ indicates the existence of a *social connection* between agents class i and j.

Definition 5 (*Social connection*) Two agents CA_1 and CA_2 are connected by a social connection according to the following rule: CA_1 is matched to the agent CA_2 if and only if there exist, some outputs of CA_1 that can match some inputs of CA_2, formally: $O_{C_{A1}} \cap I_{C_{A2}} \neq \emptyset$

Definition 6 (*Cloud Service Composition*) A Cloud services Composition β is defined by the tuple $\beta = (WS_\beta, P_\beta, QoS_\beta)$, where $WS_\beta = \{S_i | 1 \leq i \leq m\} \subseteq$ S a sub-set of selected Cloud service to form the composition and P_β define an

invoking order over WS_β. The two Cloud services S_1 and S_m represent, respectively, the start and the end node of the composition. The invocation order among these services can be in any patterns form: sequence, split, and join. β is *feasible* if it satisfies the following conditions:

$$\left\{ I^r \bigcup_{i=1}^{m-1} O_{S_i} \right\} \supseteq I_{S_{i+1}}$$

$$\left\{ \bigcup_{i=1}^{m} O_{S_i} \right\} \supseteq O^r$$

Definition 7 (*Efficient Cloud service composition*) we say a Cloud Services Composition β is *efficient* if it is feasible and optimal (the QoS_β is the best).

Definition 8 (*Dynamic Cloud Service Composition problem*) Generally, the dynamic QoS and network- Aware *Cloud* service composition problem (DCSCP) can be formally defined as a tuple DCSCP = < S, R, β, φ>, Where: S is a set of Cloud services. R represents the user query, φ the function (algorithm or approach) that compute the *efficient composition* β; $\varphi(S, R)$: $S \times R \rightarrow \beta$

Definition 9 (*local composition*) A local composition β_i^L of class agent CA_i is a composition (Definition 6) where its end node Cloud service $S_j \in CA_i.\ CWS_i$. More formally (3):

$$\beta_i^L = (S_i^L, P_i^L, QoS_i^L) \tag{3}$$

4 Cooperative Agents Based-Dynamic Cloud Service Composition

In this Section, we introduce the whole process of solving dynamic network and QoS-aware Cloud service composition problem. The core of our approach is a cooperative protocol among class agent CA. The proposed approach is based on a cooperative Multi-Agent System (MAS). Our idea is to distribute the computing of the optimal Cloud services composition among a set of cooperatives agents. Our main motivation in using MAS is to take advantage of autonomous, and self-organization agents to manage Cloud services and composition dynamicity.

4.1 Structural Description

Our approach is based on three main ideas: (i) *network coordinate system* (*GNP*) to estimate the network latency (ii) *self-organization* of agents in a social network

agent in a design-time. This allows, finding efficient Cloud service composition and manager Cloud service dynamicity in polynomial time. (iii) *Composition algorithm* to generate the optimal composite service in a reasonable time. The proposed approach is composed of four main layers (Fig. 1).

The Infrastructure Layer: This layer is used to calculate network latency by applying the GNP which will be presented. Second layer is Agents Layer; it's composed by a set of service agents, each of which handles a set of functionally equivalent Cloud services. The main role of a service agent is to ensure any changes in non-functional properties of a Cloud service. Third layer is the Composition Layer which is the main layer. It represents the service composition engine and contains two types of agents: Class Agent and Composer Agent. Finally,

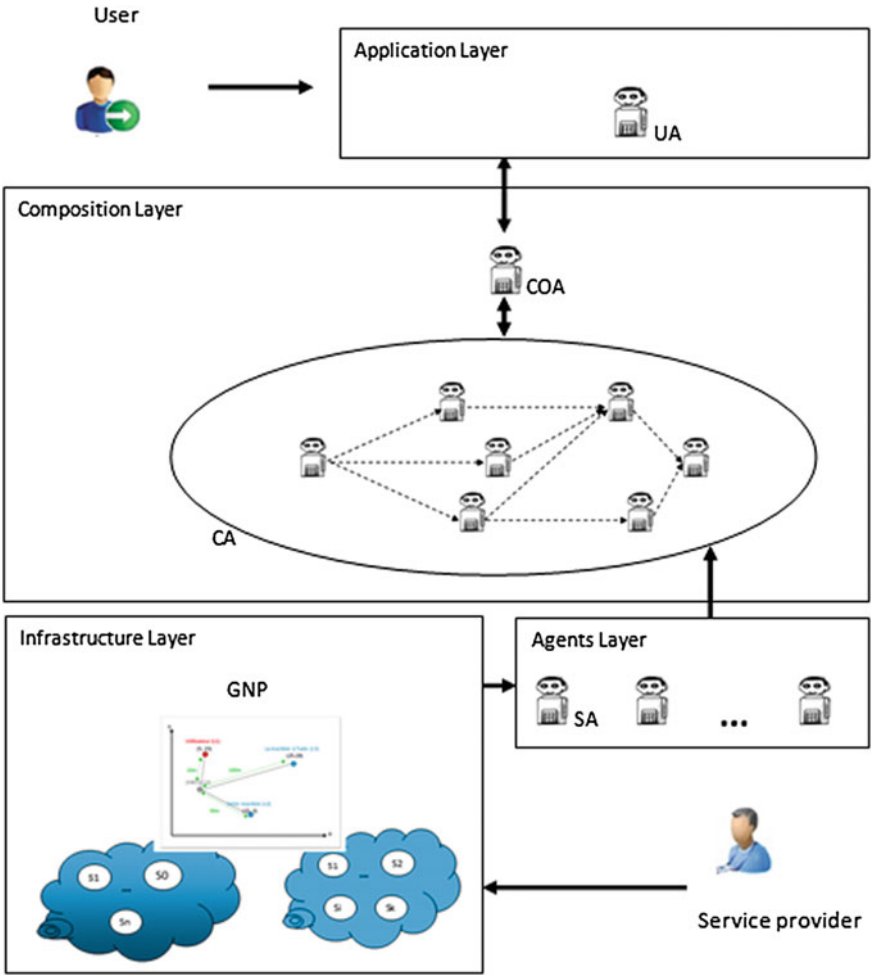


Fig. 1 Architecture

Application Layer contains the user agent and has an intermediary role between the user and the composition layer. It facilitates the communication between these two parties. Each layer is composed by a number of agents (except the infrastructure layer).

User Agent (UA): This agent has two main roles: (i) presents the user with a graphical interface allowing it to create the request. (ii) It triggers the composition process by communicating to the composer agent the user request. Service Agent (SA): This Agent is responsible of Cloud service changes. Each birth of a new service, a SA is created. Class Agent (CA): Our approach is composed by a set of n CA agents that cooperate together to compute the efficient composition. Each CA has two roles: (i) managing a service class (Definition 3) and (ii) performing its local composition (Definition 9).

Composer Agent (COA): This agent is the main entity in our approach. It initializes the composition process to fulfill the client request announced by User Agent by trigging the set of class agents to start the computing of composition process. The second role is computing the global optimal composition by selecting the best local composition sent by a set φ of class agent:$\varphi := \{CA_i | CA_i.O_{CA_i} = O^R \; and \; 1 \leq i \leq n\}$. In the next section, we describe the solution formalization of our approach.

4.2 Solution Formalization

Our composition problem is treated as a search in the service dependency graph (social network) while respecting the precedence constraints. In made, a service S_j, of CA_i agent can not be launched if all pre-conditions are fully met. That is to say that the services of an agent must wait for the arrival of the data provided by predecessors agents. To do this, choosing the best service in the CA_i agent, we must first choose an optimal service of each predecessor agent. So our goal is to determine the optimal service S_j of CA_i agent based on services (S_k) chosen from each predecessor agent. The S_j service must be the best in terms of: (i) time (run time services and network latency) and (ii) cost (cost of services and data transfer cost). All these QoS attributes are descendants attributes, that is to say low values are better, they aggregated by the sum. For each service S_j, the agent calculates the accumulated QoS value and compares it with those of the other services in his group to choose the best. This calculated value represents the optimal local composition of the agent in question. The latter, after selecting the best services, sends the value of the optimal local composition to all his successors' agents carrying out the same way. To generalize, given a CA_i agent and a service S_j such as $S_j \in CA_i$, $\forall S_k \in \text{Pred}\ (CA_i)$; $\text{j}\,\varepsilon\,\text{M} = \{1, \ldots, \text{m}\}, \text{i}\,\varepsilon\,\text{N} = \{1, \ldots, \text{n}\}$. We note (Table 1)

Table 1 Notations

Variable	Description
Te_j	Execution time of service S_j
Tl_{kj}	Latency between service S_j and its predecessor S_k
C_{S_j}	The cost of service S_j
$C_{tr_{kj}}$	The data transfert cost resulting from the execution of S_j
$\text{Size}\left(\text{tag}_{kj}\right)$	The size of data transferred between S_k and S_j
tcost	The cost per unit of data transfer for a link
x_k	A binary variable that indicates that a single service (S_k) is selected from each class agent ($x_k = 1$)

Let:

$$X_{kj} = \begin{cases} \left(Te_j + Tl_{kj}\right) * x_k & \text{if Pred}\left(CA_i\right) \neq \emptyset \\ Te_j & \text{otherwise} \end{cases} \tag{4}$$

$$Y_{kj} = \begin{cases} \left(C_{S_j} + C_{tr_{kj}}\right) * x_k & \text{if Pred}\left(CA_i\right) \neq \emptyset \\ C_{S_j} & \text{otherwise} \end{cases} \tag{5}$$

With:

$$Tl_{kj} = \sqrt{\left(abs_j - abs_k\right)^2 + \left(ord_j - ord_k\right)^2} \tag{6}$$

$$C_{tr_{kj}} = \text{size}\left(\text{tag}_{kj}\right) * \text{tcost} \tag{7}$$

(abs_j, ord_j) are the coordinates of the service S_j estimated by the GNP. X_{kj} and Y_{kj} are two intermediate variables used in the calculation of the function (f_{SC_j}) of the time, cost and local composition f_{SC_k} aggregated in service S_j. This function is given by:

$$f_{SC_j} = X_{kj} + Y_{kj} + f_{SC_k} \tag{8}$$

f_{SC_j} represents the cumulative QoS value to each service S_j. Predecessors class agents of an agent may have either the same tag or others with different tags. Two predecessors with the same tag provide the same data, so we must choose the predecessor having the minimum cumulative QoS from this list. Then, we take the maximum between the latter and the accumulated QoS services of other agents (those with different predecessors) as a service must have all its preconditions met for it to run. So there is f_1 and f_{ij}, two intermediate variables that are used to calculate respectively: (i) the optimum value of accumulated QoS forservices

belonging to the list of predecessors with the same tag and (ii) the value of overall QoS for each service S_j.

$$f_1 = \underset{S_j \epsilon (CA_i)}{Min} \left(f_{SC_j}\right) \quad \forall S_k \epsilon \, \text{Pred} \, (CA_i) \tag{9}$$

f_1 deals with list of predecessors with the same tag

$$f_{ij} = \underset{S_j \epsilon (CA_i)}{Max} \left(f_{SC_j}, f_1\right) \quad \forall S_k \epsilon \, \text{Pred} \, (CA_i) \tag{10}$$

f_{ij} deals with list of predecessors with different tags

In this case, our composition problem leads to the minimization of the function f_i given by:

$$f_i = Min\left(f_{ij}\right) \tag{11}$$

Subject to:

$$x \epsilon \{0, 1\}$$

$$\sum_{S_j \epsilon CA_i} x_j = 1 \quad \forall i \epsilon N = \{1, \ldots .., n\} \tag{12}$$

$$\text{size}(\text{tag}_{kj}) \geq 0 \tag{13}$$

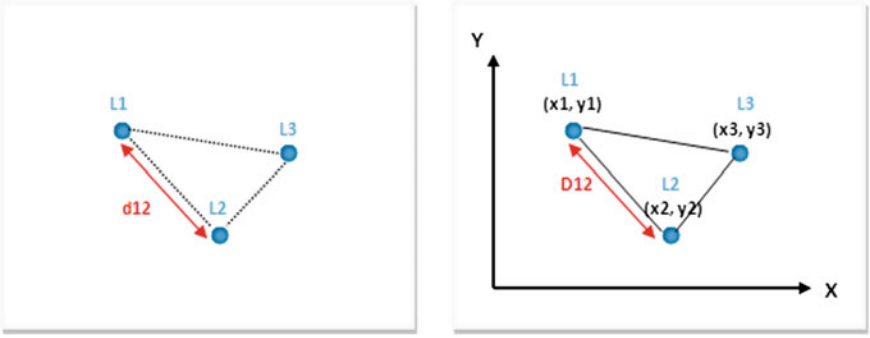
$$\text{tcost} \geq 0 \tag{14}$$

4.3 Functional Description

The composition process requires two steps: self-organization of class agent and computing the optimal composition by a cooperative protocol among agents. First, we present below the network coordinate system as we used to provide Cloud services coordinates.

Global Network Positioning (**GNP**). GNP system can calculate the distance between any two nodes in the network. It models the Internet as a 2-dimensional Euclidean space and characterizes the position of a node in the Internet by a position in this space [6]. The distance between two nodes is then provided as the geometric distance between their coordinates without explicit measures. The strategy is to set nodes "Landmarks" for references and whether they are being broadcast at any node that wants to participate in GNP. So there are two types of nodes (Fig. 2).

The landmarks: the reference points in Euclidean space. Working in a two-dimensional Euclidean space: D = 2, one must choose a minimum D + 1



a : Landmark mesureddistance b : Landmark calculateddistance

Fig. 2 Calculation of network coordinates

Landmark points and ordinary nodes: They are any other network nodes. The distance measured for points "landmarks" (Fig. 2a) is the minimum number of RTT "Round-Trip Times" using ICMP ping "Internet Control Message Protocol". The distance between landmarks i and j, Li and Lj is denoted by d_{LiLj}. Inter-Landmark Distances are transmitted to a central node (landmark). It then calculates the coordinates of the points and sends them to the Landmarks (Fig. 2b). The coordinates of N Landmarks, CL1 … CLN, are the result of minimizing the following objective function:

$$\mathrm{Fobj1}\,(c_{\mathrm{L1}}, \ldots, c_{\mathrm{LN}}) = \sum_{\mathrm{i,j}\in 1,\ldots,\mathrm{N}|\mathrm{i}>j} \varepsilon(\mathrm{d}_{\mathrm{LiLj}}, \hat{\mathrm{d}}_{\mathrm{LiLj}}) \tag{15}$$

where d_{LiLj} is the geometric distance between c_{Li} and c_{Lj}: $d_{\mathrm{LiLj}} = \mathrm{f}(\mathrm{c}_{\mathrm{Li}}, \mathrm{c}_{\mathrm{Lj}})$ and ε(.) is the error measure function (.):

$$\varepsilon\left(\mathrm{d}_{\mathrm{LiLj}}, \hat{d}_{\mathrm{LiLj}}\right) = \left(\frac{\mathrm{d}_{\mathrm{LiLj}} - \hat{d}_{\mathrm{LiLj}}}{\mathrm{d}_{\mathrm{LiLj}}}\right)^2 \tag{16}$$

Once the coordinates of Landmarks, $C_{\mathrm{L1}}, \ldots, C_{\mathrm{LN}}$ are calculated, they are transmitted with the corresponding distance function $\hat{d}_{\mathrm{LiLj}}$ to any ordinary node that wants to participate in GNP. It uses the Downhill Simplex algorithm [30] to solve the optimization problem.

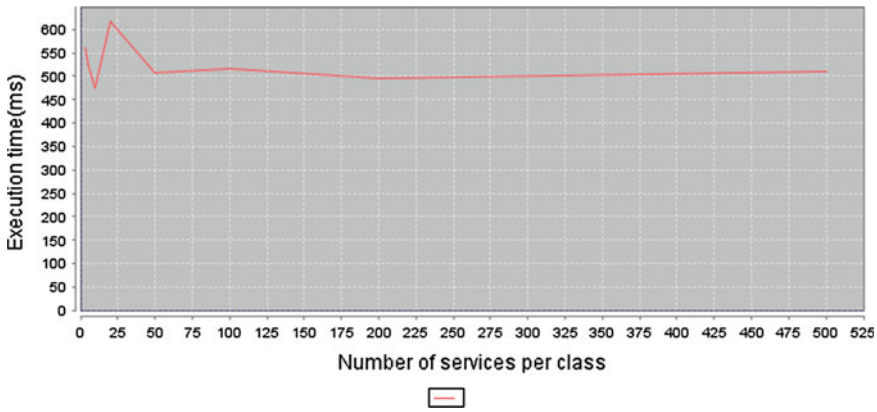
Self-organization of Class Agent Agents are organized as a social network agent in the design-time. This is, in order to accelerate the time of computing the optimal composition. As defined in Definition 3, class agents are linked based on Input and Output of their services. The self-organization processes beyond the scope of this work.

Computing Optimal Cloud Services Composition In run-time, class agent and composer agent cooperate together to generate the efficient composition. This step is based on computing a set of local composition (Definition 9) by class agent and the global composition by the composer agent. Computing the local composition: each class agent maintains a list of all his predecessors' class agents at design-time. It sorts out this list based on the data they provide (tag) to the agents providing the same data in contiguous spaces. Then, it calculates the cumulative QoS to each of its services based on selected services of each predecessor, and selects the best. For lists of the same tag predecessors, it selects the minimum value and then, for the rest it selects the maximum value because each agent awaits the results of all his predecessors. After calculating the cumulative values optimal QoS for each service it manages, the CA_i agent selects the service with the best cumulative QoS value (minimal). Computing the global composition: The global Cloud service composition process is based on the flowing cooperative protocol:

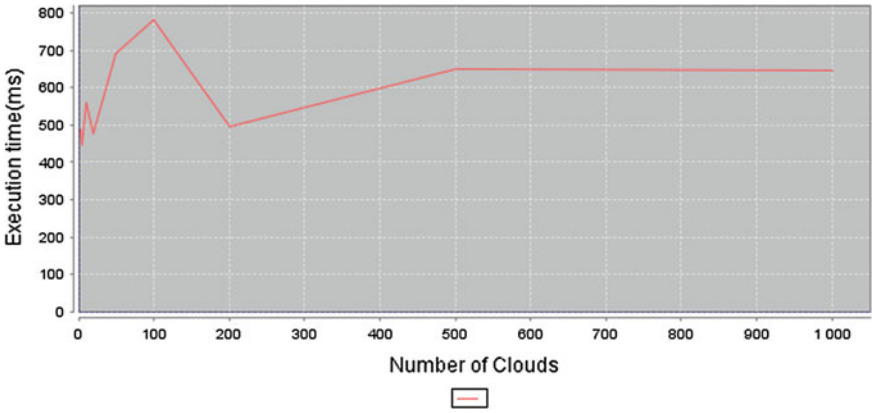
1. The UA constructs and sends the query R to the composer agent.
2. The COA builds the set θ of Class agents that their Input I_{CA} matches the user query Input R^I: $\theta = \{ CA_i | I_{CA_i} \cap I^R \neq \emptyset \}$. The composer agent checks whether the set θ is empty. If it is true, then notify the user that its query R cannot be achieved. Otherwise, it sends the message *request* (ID_{cA}, R) to the each agent $CA_i \in \theta$.
3. If class agent CA_i receives the message *request*, then it computes its local composition (or sub-composition)$\beta_i^L = (S_i^L, P_i^L, QoS_i^L)$. In this step, $S_i^L = \{S_1 | S_1 \in \mathrm{CA_i.CWS_i}\}$ is the best Cloud service from the set of services CWS_i in the class managed by CA_i. Then, it sends the message *MyComposition* (CA_i, β_i^L, R) to all successors agent based on its social network.
4. If the class agent receives a set of *MyComposition* message, it computes its local composition. If the output of its class matches the output of the user query, then sends its composition to the composer agent. Else, it sends its local composition to its successor and Go to 3).
5. Finally, the composer agent computes the global composition and sends it to the user agent which forwards it to the end user through the interface of the application layer.

5 Experimental Results

To implement our system, we use the Multi-Agents platform JADE with Eclipse framework. The resulting graphs are generated using java library JFreeChart. We have created n agent class, m Cloud services shared on all agent class and k Clouds containing services. The execution time values and cost of services are generated randomly. We evaluate the performance of our approach using several configurations. The results of the first configuration are represented by the Fig. 3a.



a : Execution time per number of services per class



b : Execution time per number of Clouds

Fig. 3 Tests of our approach

We can deduce that by increasing the number of services per class, the execution time varies at the beginning and then stabilizes. In Fig. 3a, it varies between 470 and 620 ms. From a 50 number of services per class is almost constant (500 ms). In the second configuration, the results are represented by the Fig. 3b. Similarly, when we increase the number of Cloud; the execution time varies at the beginning and then stabilizes. Note that the computing time in the same class agent is negligible compared to the overall time, and the change in the number of services per class or number of cloud containing the services does not affect much the overall execution time since the structure of the social network of agents has not changed.

In contrast to previous figures, we see that in Fig. 4 the overall execution time grows with the number of class agents. Made in, when increasing the number of agents, added nodes of the social network of agents and the size of the search graph

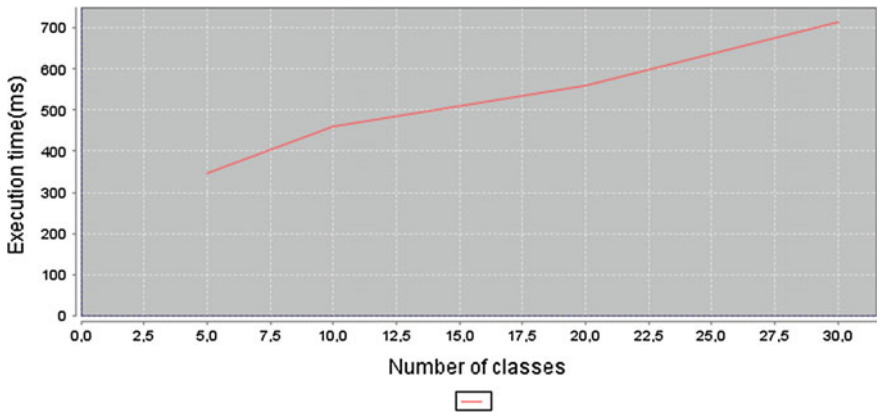


Fig. 4 Execution time of our approach per number of Classes

increases. Therefore the problem size increases, but the overall calculation time increases slowly. Thus, our algorithm has an ability to be adapt to the increased size of the problem, and the results prove its performance and scalability.

6 Conclusion and Future Works

In our present work, we propose an efficient Cloud service composition approach based on Multi-Agents System. Our solution is based on three ideas: (i) Network Coordinate System, named GNP, used to estimate accurately the coordinates of Cloud services, is to calculate an estimation of the network latency, (ii) Self-organization of agents as a social network (Each class agent managed a set of services sharing the same functional proprieties.) This is, in order to minimize the time computing and distribute the Cloud services composition computing among class agents. (iii) Computing the optimal composition based on the protocol presented above. This approach is characterized by accuracy and reacts to dynamicity of Cloud services. Our approach is scalable while keeping the constraint of minimization the QoS. In our future works, we will plan to use Case-based reasoning (CBR) concept to improve our approach.

References

1. Amazon S3. Amazon simple storage service cloud computing platform: http://aws.amazon.com/s3/
2. Windows Azure Platform. Microsoft cloud computing platform: http://www.microsoft.com/windowsazure/
3. http://www.ibm.com/us/en/

4. Ye, Z., Zhou, X., Bouguettaya, A.: Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing. Springer, Berlin (2011)
5. Pisinger, D.: Algorithms for Knapsack Problems. PhD thesis, University of Copenhagen, Department of Computer Science (1995)
6. Ng, T.S., Zhang, H.: Towards Global Network Positioning. New York USA (2001)
7. Klein, A., Ishikawa, F., Honiden, S.: Towards network-aware service composition in the cloud. In: WWW 2012, Lyon, France, 16–20 Apr 2012
8. Klein, A., Ishikawa, F., Honiden, S.: A scalable distributed architecture for network- and QoS-aware service composition. In: IJAWS 2012, Japan (2012)
9. Zou, G., Chen, Y., Yang, X., Huang, R., Xu, Y.: «AI planning and combinatorial optimization for web service composition in cloud computing. In: CCV Conference 2010, Singapore, 17–18 May 2010
10. Gutierrez-Garcia, J.O., Sim, K.-M.: Self-Organizing Agents for Service Composition in Cloud Computing. doi:10.1109/CloudCom.2010.10
11. Gutierrez-Garcia, J.O., Sim, K.-M.: «Agent-based service composition in cloud computing». In: Kim, T.-h., et al. (eds.) GDC/CA 2010, CCIS 121, pp. 1–10 (2010)
12. Zhou, J., Athukorala, K., Gilman, E., Riekki, J., Ylianttila, M.: Cloud architecture for dynamic service composition. Int. J. Grid High Perform. Comput. **4**(2), 17–31 (2012)

Distributed Computing Architecture on Epiphany MIMD Accelerators

Łukasz Faber

Abstract Last few years have seen introduction of more and more advanced manycore processors. Both very well known devices like GPGPU and Intel MIC and less popular, but still very interesting, like Epiphany. There is also a growing popularity of cheap, credit-card-sized devices offering advanced features and high computational power. One of this kind of devices is the Parallella board that focuses on the parallel computing and features the Epiphany coprocessor. In this paper we propose an architecture for distributed computational systems based on Parallella and the "multiple instruction, multiple data" (MIMD) coprocessor Epiphany. This manycore processor consists of sixteen cores connected by a mesh network-on-a-chip. The presented architecture enables the usage of multiple Parallella boards in a single system with a possibility to also use other computing units. The target usage of this system are multi-agent systems (MAS) and we present selected scenarios that could be easily implemented and would benefit from the properties provided by multiple MIMD devices.

Keywords Distributed computing · Parallel computing · Epiphany · Multi-agent systems

1 Introduction

There are several available solutions offering the manycore highly parallel programming environments, for example, GPGPU (General-purpose computing on graphics processing units) devices with CUDA and OpenCL [1] or Intel Xeon Phi (Intel MIC) [2]. However, even with these devices becoming faster with each new product, there are still attempts to use multiple instances of them in a single,

Ł. Faber (✉)
AGH University of Science and Technology, Kraków, Poland
e-mail: faber@agh.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_10

distributed architecture. There are many reasons for this approach: costs [3] (more advanced versions of GPGPUs and Xeon Phi usually cost much more than simpler ones), requiring more computational power than a single device can provide [4], etc.

On the other hand, there is a growing popularity of the small integrated, power consumption- and cost-oriented computing boards (for example, Raspberry Pi). Few years ago, the manycore approach was successfully implemented on such a simple board—Parallella[1] [5] created by Adapteva. It is a small (credit card-sized) board comprising of 16-core Epiphany coprocessor and the main dual-core ARM processor.

Such boards are interesting for researchers due to their low costs, simplicity, and low level requirements for starting to work with them. Parallella has its benefits of being a standalone, plug-and-play type of a coprocessor similar to Intel Xeon Phi. "Standalone" in case of Parallella means that it's a completely separate computer unit that can run independently of any other nodes. On the other hand, it's plug-and-play, because all it takes, to connect to it, is attaching an Ethernet cable.

We have been working on using multiple Parallella boards for more advanced computation and, in this paper, we want to present an architecture of the system that uses multiple devices to provide a computational environment for multi-agent systems. The proposed architecture is distributed, scalable and flexible as it depends solely on connecting boards as needed using the Ethernet network. We also show the advantages gained by the MIMD nature of the Epiphany coprocessor (in comparison to mostly SIMD GPGPUs). Thanks to this solution we can gain a cheap, highly-parallel and distributed environment that can be created ad hoc in any place that has an Ethernet network.

In the following sections, we briefly introduce the Parallella platform and Epiphany manycore processor (Sect. 2), its memory and programming models. Then, in Sect. 3, we present the logical architecture of the system. In Sect. 4 we discuss the implementation extensively. In Sect. 5 we present the sample scenario that can be run in the system. And, finally, we discuss the consequences and future steps in Sect. 6.

2 Parallella

Parallella is a hardware platform built on top of the manycore Epiphany [5] coprocessor, created by Adapteva in 2011. It was funded by the Kickstart campaign.[2] The board was first presented in June 2014.

The Epiphany processor consists of a 2D array of nodes (so-called "eNodes") connected by a mesh network-on-a-chip. Each node consists of a single RISC core

[1] https://www.parallella.org/.

[2] https://www.kickstarter.com/projects/adapteva/parallella-a-supercomputer-for-everyone.

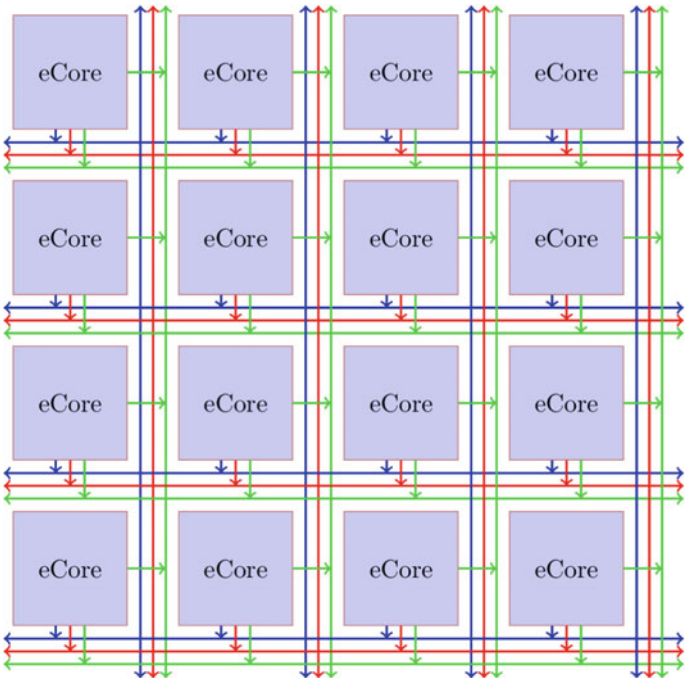


Fig. 1 eMesh Network-on-a-Chip. The *blue lines* are cMesh (used for on-chip writes), *green*—xMesh (off-chip writes), *red*—rMesh (read requests)

(called "eCore"), a DMA engine, 32 kB of memory and network interface. Each core includes a 32-bit floating point RISC CPU, local memory, a DMA engine, an event monitor and a network interface. The mesh connections on a 16-core processor are shown in Fig. 1. Each "eCore" contains a floating-point unit (FPU), an arithmetic logic unit (ALU) and a 64-word register file, as shown in Fig. 2.

The address space in Epiphany is flat and consists of 2^{32} bytes. Each node has 32 kB of its own local range of memory. However, memory of each node can be accessed by prefixing the address with the globally addressable ID.

eMesh architecture has two important properties: writes are preferred over reads —for a single read there need to be two transactions: one for a read request and one for an answer, and non-local memory accesses are weakly ordered.

Main goals of the Epiphany architecture are: power efficiency (a single 16-core Epiphany processor consumes maximum 2 W, the whole Parallella board requires around 5 W), scalability, easy programming model, high performance (2 GFLOPS per single core).

Currently, there are 16- and 64-cores Epiphany processors available. However, 64-core versions were only produced in limited numbers and not reachable through any other way than directly from Adapteva.

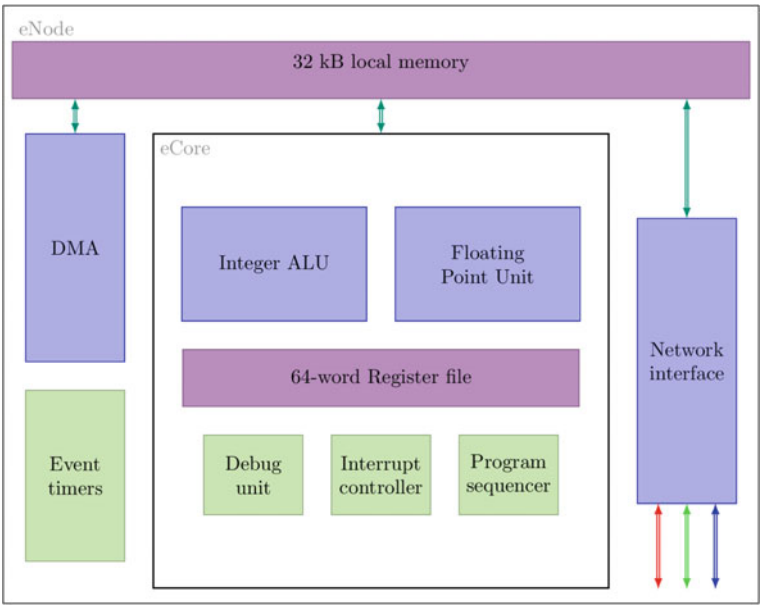


Fig. 2 eNode components. Each eNode has an eCore and 32 kB of local memory, network router, DMA engine and two event timers. Each eCore has a 64-word register file, FPU, ALU, interrupt controller, sequencer and debug unit

The Parallella board uses a 16-core Epiphany processor (E16G301), a Xillinx Zynq (models 7010 or 7020 with two ARM cores) and 1 GB of RAM. Additional components include: 1 Gbit Ethernet interface, USB and HDMI ports, MicroSD slot. The standard operating system (in this case—Linux) boots from the MicroSD card onto ARM processor and can communicate with the Epiphany using an e-Link interface. 32 MB of memory are shared between ARM processors (host) and Epiphany eCores can use it in the same way as internal memory.

The memory size usable for the programmer is dependent on many factors. In the most common linker configuration the internal memory (32 kB) contains: program code, global variables, stack. And the fragment of the external memory (32 MB) is used for the C standard library code, data and stack [6].

Programming on the Epiphany side is done in a usual way. The SDK supports the standard C library with mathematics functions. The "workgroup" concept is supported and each of created workgroups can have different code loaded. The important part to remember is that Epiphany is a MIMD processor and each core can execute completely different code. Alternative approaches to using the basic SDK are MPI [7] and OpenMP [8].

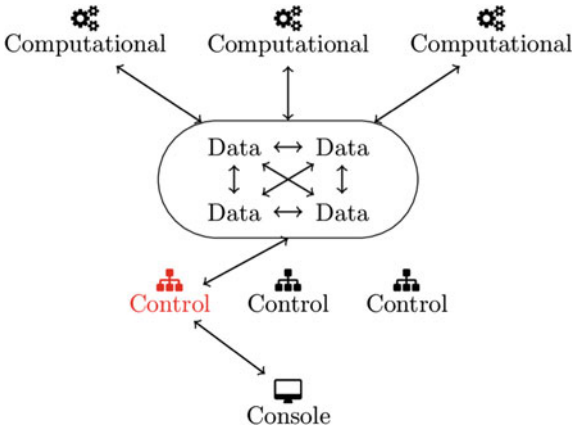


Fig. 3 Logical architecture of the proposed system. Four node roles are visible: computational, data, control, console. The *arrows* shows the connectivity between active nodes. All computational and data nodes are always active. There is only one (*red*) control node active (active-passive model). There is only one console node in the system

3 Architecture

There are four logical node types defined in the system, as shown in Fig. 3. Each of them has a specific role and tasks to perform in the system:

1. Data nodes provide a backing distributed data storage.
2. Control nodes provide management over other nodes. They perform command requests to computational nodes then collect results. Moreover, they also need to split tasks for computational nodes on the basis of the task definition.
3. Computational nodes perform the real work in the system by executing the algorithms. In the presented case we use only Parallella-based computational nodes. Each logical unit is mapped directly to a single Parallella board and a single Epiphany processor. However, in the later stages of the project we plan to add more computational types of units. Especially full JVM-based units.
4. Console nodes are I/O units providing code to execute and collecting results.

There is nothing preventing a single physical node from performing all roles. However, Parallella has a limited computational resources (especially memory) and that may create a problem with sharing computational, data and control roles.

The system also introduces a term "workgroup" that describes any number of adjacent Epiphany cores. This is mainly for easier exploitation of MIMD nature, as each workgroup can execute different code. Workgroups can share any data.

4 Implementation

The current version of the system was implemented in Java 8, as it is the language most fitting our purposes. The work was centered on four areas: network communication, node implementation, interface to Epiphany, and a way to define tasks. The general overview of how nodes are connected is shown in Fig. 4.

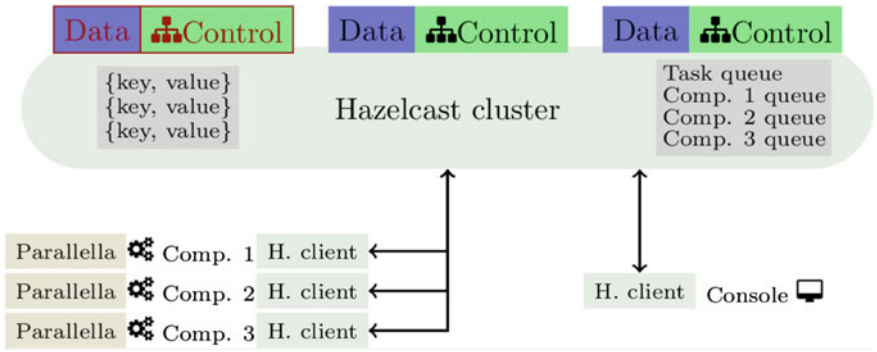


Fig. 4 Implementation of the system using Hazelcast. Data/control nodes are connected in a single Hazelcast cluster and hold distributed maps and queues. Only one control node is active but any can become active in case the current one fail. The console and computational nodes are implemented as clients connecting to the cluster

4.1 Network

For network communication we used Hazelcast,[3], an in-memory data grid. It provides a large set of distributed data structures and mechanisms, for example: distributed maps, queues, topics, locks, executor services. Moreover, it can be adapted to most network configurations. By default it performs service discovery and communication using multicast. However it can also use a central registry or direct IP addressing (usually for non-local networks).

In our system we required the following communication channels:

- a command request channel from a console node to control nodes,
- a command request channels from control nodes to computational nodes,
- results channels from computational nodes to control nodes,
- results channels from control nodes to computational nodes.

We have implemented command request channels as queues (Hazelcast's `IQueue`). There is one task queue for sending tasks from console and it is handled by the active control node. Then, there is a single task queue created for every computational node in the system.

For the results channels we decided to use a distributed map (key-value store). This gives us flexibility to access results by any node at any time.

[3] http://hazelcast.org/.

4.2 Node Implementation

We implemented separate console and computational nodes and a single application for data and control roles. This decision was directly influenced by the facilities offered by Hazelcast and the aforementioned fact, that the Parallella board has low computational resources.

Console and computational nodes are implemented as Hazelcast clients. It means that they do not participate in the data cluster—they do not have any copy of data. However, it does not mean that it is impossible for them to access data and use communication facilities. They just need to connect (transparently) to any of data nodes to use them. It is easy to see why this decision has merit for the console node: it is started only for sending commands or receiving results and is should execute quickly. It would be counter-productive for it to connect to all other nodes and synchronize data with them. On the other hand, the fact that we decided not to attach the Parallella nodes to the cluster may look counter-intuitive—these nodes are running all the time. Our decision was dictated by the small RAM size available on Parallella. However it would be possible and worthy of further tests to fine-tune the Hazelcast configuration in order to have a Parallella node to have a local copy only of required data.

Data and control nodes were implemented as a single application and are fully-participating Hazelcast cluster nodes.

It is required for at least a single Parallella (computational) node and a single data/control node to exist in the system.

The way the system handles changes in nodes presence is dependent on the node types:

- For data nodes appearance is automatically handled by Hazelcast. Node disappearance is also automatically handled and no data is lost, however, the failure resiliency is dependent on the configuration.
- Control nodes need to elect the active node. It is easily done using Hazelcast-provided identifiers. In case of the active node disappearing the election is performed again. If there is no control nodes left, the system is dysfunctional.
- Computational nodes appearance and disappearance is detected by all control nodes using Hazelcast-provided events and services.

4.3 Epiphany Interface

The main problem to solve was to make a usable interface to the Epiphany processor and provide a way to push data and code to it. We have reviewed several choices regarding a way to access the Epiphany HAL library. We considered using,

for example, OpenCL implementation,[4] JNI (Java Native Interface), JNA (Java Native Access).[5] We have chosen the last one mainly due to flexibility. Although each of these methods has its own problems:

- **OpenCL** although general Java OpenCL support is good (for example, [9]), this particular implementation for Epiphany is not supported by any of known bindings. It was the main problem, as we wanted to provide as seamless integration as possible.
- **JNI**—it seemed to be a better approach than using (more universal) OpenCL. It would provide us a direct interface to the library. However, using it requires preparing a lot of the "glue" code.
- **JNA**—in short, it is a wrapper around JNI that basically removes the need to write glue code (it is generated automatically) but loses performance.

As we did not require a high performance of the bridging code and wanted to keep things simple, we used JNA. This way we have one-to-one mapping of the HAL library for Epiphany to the Java code.

We also created a wrapper around the Epiphany C compiler (e-gcc). This was required in order for nodes to compile incoming C source code by themselves. It is possible to compile it once for all nodes but it would be less flexible, as all nodes would be required to have an identical configuration and the SDK version.

4.4 Task Definition

The most basic task must provide answers for the following questions:

1. How many and what kind of units to use?
2. How should be they grouped together?
3. What sub-task should each of the group run?
4. What is the input and the output?

In an answer for the first question we need to provide a info whether we want to use the full JVM node or Epiphany cores. And, in the second one, how they will be working together. It is especially important for Epiphany cores, as the corresponding workgroups need to be created. As for now we do not allow grouping together units of different type (JVM and Epiphany).

The answer to the third question requires us to provide the code to run: a Java class (for JVM nodes) or an Epiphany C source code file (for Epiphany nodes). It is a single code for each unit group (workgroup).

[4]There is an OpenCL library targeting Epiphany: http://www.browndeertechnology.com/coprthr.htm.

[5]Available at: https://github.com/java-native-access/jna.

For the last one, we need to provide input data—both contents and target variables—and define which values should be collected after the successful run. For Epiphany units it will be described by the memory address, the values to copy and the size of the target variable.

5 Sample Scenarios

The very obvious scenarios matching this architecture are all island-based evolutionary algorithms that do not require the global knowledge. For example, Evolutionary Multi-Agent Systems (EMAS) which merge evolutionary algorithms with a multi-agent system [10]. In these systems agents represent individuals that are subject to evolutionary processes. Agent populations are located on separate islands and there is no need for the global knowledge or synchronization. When needed, agents can migrate to other islands. One of advantages of such model is easier distribution of the implementations.

This scenarios perfectly matches the architecture, as we can model each population (island) on a single Parallella board. Depending on the needs of the particular algorithm, we may implement agents in vertical or horizontal fashion.

- **Vertical implementation** makes use of the MIMD nature of Parallella—each agent is completely separate entity (or code) running on a separate core. Agents still can communicate with each other.
- **Horizontal implementation** treats the Parallella more like a stream processor where all cores split a large population into smaller chunks and process them in SIMD fashion.

Another scenario we are planning to implement and use with the presented architecture are hybrid particle-MAS simulations [11]. Dissipative particle dynamics simulations benefit from highly parallel environments and, in case of Parallella, they can co-execute with agent simulations on a single processor sharing the memory. Separate boards can be used for simulating adjacent parts (sections) of a blood vessel.

6 Conclusions and Future Work

In this paper, we have presented the initial version of a distributed computational system based on Ethernet-connected “multiple instruction multiple data” (MIMD) Epiphany processors. We wanted to show that it is perfectly reasonable to build a complex system comprising of multiple small, independent, highly parallel units. The implementation has particular benefits related to the high speed local memory access of Epiphany cores [12]. In case when it is possible to split the computation

into independent parts that communicate only in sparse synchronization points, it can be easily mapped to this architecture.

This system benefits from the MIMD-nature of the Epiphany processor and makes it possible to multiple small independent (but sharing memory) agents on a single board. However, it does not stop one from writing the processing in another, for example, stream-like way.

The initial version of the presented work has some shortages that can and will be improved in later stages. For example, the implementation lacks the full JVM node version and the task definition is very imperative and verbose. Another problem is the fact that Epiphany code needs to be written in C. It could be beneficial not only for our project to provide a way to use a Java syntax for writing the coprocessor code.

Our next plans is to profile our system, based on the aforementioned scenarios, in order to test our assumptions regarding the performance of the system.

Acknowledgments The research reported in the paper was supported by the grant "Hybrid model of the early detection of internal diseases based on the paradigm of interacting particles and multi-agent system" (No. DEC-2013/09/N/ST6/01011) from the Polish National Science Centre.

References

1. Stone, J.E., Gohara, D., Shi, G.: Opencl: a parallel programming standard for heterogeneous computing systems. Comput. Sci. Eng. **12**(3), 66–73 (2010)
2. Chrysos, G.: Intel Xeon Phi TM Coprocessor-the Architecture. Intel Whitepaper (2014)
3. Harding, S.L., Banzhaf, W.: Distributed genetic programming on GPUs using CUDA. In: Workshop on Parallel Architectures and Bioinspired Algorithms. Raleigh, USA. Citeseer (2009)
4. Heinecke, A., Vaidyanathan, K., Smelyanskiy, M., Kobotov, A., Dubtsov, R., Henry, G., Shet, A.G., Chrysos, G., Dubey, P.: Design and implementation of the linpack benchmark for single and multi-node systems based on Intel Xeon Phi coprocessor. In: IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS), pp. 126–137 (May 2013)
5. Olofsson, A., Nordström, T., Ul-Abdin, Z.: Kickstarting High-performance Energy-efficient Manycore Architectures with Epiphany, 2–5 Nov 2014
6. Epihany SDK Reference, rev. 5.13.09.10, http://adapteva.com/docs/epiphany_sdk_ref.pdf
7. Ross, J.A., Richie, D.A., Park, S.J., Shires, D.R.: Parallel programming model for the epiphany many-core coprocessor using threaded MPI. In: Proceedings of the 3rd International Workshop on Many-core Embedded Systems, pp. 41–47. ACM (2015)
8. Papadogiannakis, A., Agathos, S.N., Dimakopoulos, V.V.: OpenMP: Heterogenous execution and data movements. In: Proceedings of 11th International Workshop on OpenMP, IWOMP 2015, Aachen, Germany, 1–2 Oct 2015, chap. OpenMP 4.0 Device Support in the OMPi Compiler, pp. 202–216. Springer International Publishing, Cham (2015), http://dx.doi.org/10.1007/978-3-319-24595-9_15
9. Pratt-Szeliga, P.C., Fawcett, J.W., Welch, R.D.: Rootbeer: seamlessly using GPUs from Java. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), pp. 375–380 (2012)

10. Byrski, A., Dreżewski, R., Siwik, L., Kisiel-Dorohinicki, M.: Evolutionary multi-agent systems. Knowl. Eng. Rev. **30**(02), 171–186 (Mar 2015), http://www.journals.cambridge.org/abstract_S0269888914000289
11. Faber, Ł., Boryczko, K., Kisiel-Dorohinicki, M.: Hybrid architecture for simulation of blood flow with foreign bodies. In: ECMS, pp. 523–529. Brescia, Italy (2014), http://www.scs-europe.net/dlib/2014/ecms14papers/dis_ECMS2014_0138.pdf
12. Varghese, A., Edwards, B., Mitra, G., Rendell, A.P.: Programming the Adapteva Epiphany 64-core network-on-chip coprocessor. Int. J. High Perform. Comput. Appl. (2015)

A Fail-Safe NVRAM Based Mechanism for Efficient Creation and Recovery of Data Copies in Parallel MPI Applications

Artur Malinowski, Paweł Czarnul, Maciej Maciejewski and Paweł Skowron

Abstract The paper presents a fail-safe NVRAM based mechanism for creation and recovery of data copies during parallel MPI application runtime. Specifically, we target a cluster environment in which each node has an NVRAM installed in it. Our previously developed extension to the MPI I/O API can take advantage of NVRAM regions in order to provide an NVRAM based cache like mechanism to significantly speed up I/O operations and allow to preload large files and operate efficiently on them. In this work, we show how to provide fail safe data write to such files using NVRAM and how to recover from failures. This provides an efficient alternative to costly checkpointing provided an application can store its consistent state in a file.

Keywords NVRAM · Parallel MPI I/O extension · Fail safe · Distributed cache

The research in the paper was supported by a Grant from Intel Technology Poland.

A. Malinowski (✉) · P. Czarnul
Department of Computer Architecture, Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
e-mail: artur.malinowski@pg.gda.pl

P. Czarnul
e-mail: pczarnul@eti.pg.gda.pl

M. Maciejewski · P. Skowron
Intel Technology Poland Sp. z o.o, Gdansk, Poland
e-mail: maciej.maciejewski@intel.com

P. Skowron
e-mail: pawel.skowron@intel.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_11

1 Introduction

Nowadays, high performance computations (HPC) typically run on clusters that consist of hundreds or thousands of nodes or more for a total of millions of cores according to the top500.org list. In such huge environments, failures are inevitable and application designers should reduce a resulting negative impact. One popular solution to this problem is to use checkpointing, that is usually composed of three steps:

1. Stop/synchronize application execution.
2. Copy the data into storage/non-volatile media.
3. Continue execution.

Unfortunately, for applications that operate on large data, copying the data can consume a significant amount of time—some studies prove, that even more than 50 % [1]. Another drawback of checkpointing is usually connected to the overhead of a development process [2]. Even though researchers proposed some automated solutions that are even transparent to programmers, these usually require compatibility of environment components and additional configuration.

With smaller cluster environments, when the probability of a hardware failure is lower, many applications do not include checkpointing at all. In many cases, execution time overhead and the requirement for an additional development effort prevail over benefits from system fail-safeness. Not all applications require a high level of fault tolerance. However, we believe that if disadvantages of introducing protection against some type of failures were reduced, such solution would be really useful.

In the report based on data collected for over 9 years at Los Alamos National Laboratory, Schroeder and Gibson distinguished five root causes of HPC systems failures: hardware, software, network, environment and human [3] induced. Although the hardware issues were reported to be the most frequent, protection against this type of failures usually involves incorporation of expensive checkpointing.

Prevention from data being damaged after a failure requires persistent storage. In most cases, SSD or HDD file systems are used, but development of new storage techniques is already underway. From the perspective of this article, an interesting alternative for classical storage devices is Non-volatile Random Access Memory (NVRAM). Hardware components with performance at the level between RAM and modern SSD, with data persistence and byte level access, are expected to be released soon [4]. We believe, that NVRAM devices could be utilized in HPC environments in order to provide better fault tolerance.

A comprehensive HPC solution requires not only a sophisticated hardware, but also software adapted to the hardware features. The Message Passing Interface (MPI) is a widely used API that helps create HPC distributed applications for multi-node clusters. A wide range of functions and availability of several stable implementations make it a reasonable choice not only for development, but also as a popular subject of research on fail-safeness in HPC. Within this paper, we present an easy-applicable and a low overhead MPI solution, with the use of NVRAM, that allows to recover from software, some hardware, network and environment caused failures.

2 Related Work

As checkpointing is probably the most popular method of providing fault-tolerance to MPI applications, many research papers focus on its improvements. An interesting example is Berkeley Lab Checkpoint/Restart (BLCR) platform [5, 6]. The library aims to be transparent to the developer, have a wide application/system coverage, and should not significantly increase execution time of application. Although the implementation fulfilled those requirements, we rejected this idea mainly because of the overheads related to creating a checkpoint.

One of the main features of NVRAM is persistence and it was also used in many checkpointing improvements [7–9]. Despite the fact that NVRAM is expected to outperform classical storage, the level of additional execution time for preparing a checkpoint could be still unsatisfying. In [10] we presented how NVRAM can be used within implementation of MPI one sided I/O to be used for checkpointing.

Apart from checkpointing, several methods for fault-tolerant applications, written using MPI, were proposed. In 2002, Gropp and Lusk [11] distinguished and described in details four different techniques. The list of approaches, together with the comparison to the solution described in this paper, is as follows:

1. Checkpointing—described in the previous paragraph.
2. Using intercommunicators to reduce the failure impact—applicable only for worker/manager applications, requires usage of dynamic process management.
3. Modification of MPI API semantics in order to provide fault tolerance—such proposal is unacceptable in generic solutions, because it could make most of existing applications incompatible with modified MPI.
4. Extending MPI specification—rejected because of the need for using an additional set of functions that would complicate development process.

Our solution may seem to be similar to the third approach, but the semantics of API functions stay untouched—the proposed solution is only an additional guarantee of a consistency of a file being accessed.

Another mechanism was investigated by Fagg and Dongarra [12]. Their solution, called FT-MPI, focused not on data recovery after the application has crashed, but on rebuilding an application structure at runtime. To achieve this goal, they introduced several additional states of communicators together with different handling strategies. Performance results were promising, however, the implementation is not popular, probably because of incompatibility with the MPI standard.

3 Proposed Solution

We first refer to an MPI I/O extension using NVRAM that we developed previously and then propose a new fail-safe mode of this solution.

MPI I/O supported by NVRAM cache

In our previous research we provided MPI I/O extension[1] based on in-system distributed cache with data located in NVRAM [13]. The extension is transparent to the developer, because the API stayed unmodified. The main goal of the research was not only to increase input and output operations performance, but also to ease the development process by making access of small data chunks reasonably fast (a programmer can avoid implementation of data staging, buffering, etc.). In the extension, each computing node is equipped with its own NVRAM device. An accessed file is split into continuous parts managed by cache managers and pre-fetched into NVRAM regions in the initialization phase. MPI I/O calls, such as write and reads, operate on NVRAM regions, either from the same or other nodes.

The extension, tested using three applications, obtained better performance than the regular MPI I/O if only certain criteria were fulfilled:

1. Application was running long enough to compensate the overhead for cache initialization and deinitialization.
2. Application was data intensive and accessed data chunks were small.

Fail-safe mode

NVRAM cache described and tested in the previous research was also persistent, but did not ensure file consistency, therefore it could not be considered as fail-safe. This section is an original extension of a previous research. Many data-intensive applications operate on a file. The proposed extension provides a mechanism, that keeps a file in a consistent state at the level of a single write operation. The state could also be recovered in case of an application's crash. To benefit from the solution, it is required that an application stores its current state in a file. In case of an error, all pending write requests are buffered and can be completed in the provided fail recovery routines after application rerun. The application should also be able to start computations from its previous state. Table 1 shows main differences between using typical checkpointing mechanisms and the proposed solution. At the application level, checkpointing is a more general approach that can handle messages in transit as an element of an application state. Our approach requires a consistent state written to a file which is often fulfilled e.g. geometric SPMD applications or when an application operates on many files such as in image processing [14]. In a wider context, it can also be used during workflow execution [15] for computational MPI based services.

The fail-safe mode ensures consistency by buffering data in a file on an NVRAM device before writing into a cache. If the write procedure ends with success, the recovery buffer is removed. Otherwise, it is possible to reopen the cache and retry the write operation. Figure 1 presents the lifecycle of the cache and the recovery buffer. Recovery process proceeds as follows: initialize cache structures as when

[1] https://github.com/pmem/mpi-pmem-ext/tree/master/mpiio_extension2.

Table 1 Differences between checkpointing and fail-safe mode of MPI I/O NVRAM extension

	Checkpointing	NVRAM fail-safe mode
Environment requirements	Optional additional library	NVRAM + MPI I/O extension
Application requirements	Development overhead	Consistent state stored in a file
Time overhead	Noticeable	Negligible
Fault tolerance	All failure causes	All failure causes except hardware

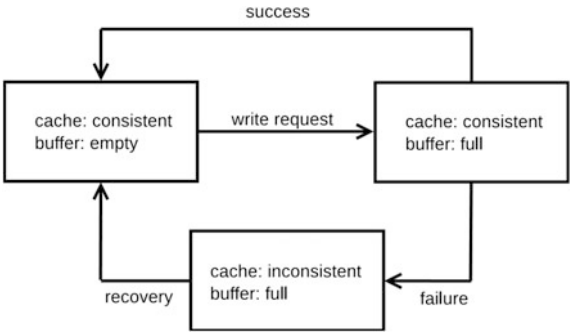


Fig. 1 Diagram of possible cache and recovery buffer states

opening a file, but using existing NVRAM range instead of new memory allocation; for each unfinished write request stored in file, try performing it again.

In the optimized version, at the phase of cache initialization, a buffer—single file with preallocated space, is created to store initiated write operations. Figure 2 presents the structure of a buffer with an activity flag (which informs whether an operation stored in the buffer is not completed), size and offset of data within the whole file, and data itself. If the size of a processed data is equal or lower than the buffer capacity, instead of new file creation the buffer is used. To ensure that data is synchronized, fast pmem_memcpy_persist operation from the libpmem library is used. Eventually, the buffer is removed in a cache deinitialization phase.

The fail-safe mode is not only beneficial in a case of write operations. Although initialization of NVRAM cache can consume a lot of time due to prefetching of data, recreation of a cache after a failure is a relatively cheap operation. This advantage, together with insignificant execution time overhead, leads to the conclusion, that usage of a fail-safe extension is also recommended for files opened in read-only mode. The solution completed successfully a set of functional tests, where the application was stopped at random moments in time.

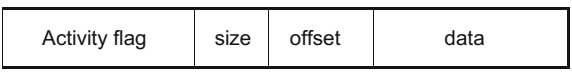


Fig. 2 Structure of a write request buffer. Meta-data is kept minimal

4 Experiments

Experiments are focused on showing fail-safe mode overhead, scalability of the solution and recovery time. The aim of overhead measurement was to compare the proposed solution to the unmodified and unextended MPI I/O with no checkpointing (further called "regular"). Results of NVRAM cache were obtained including fail-safe mode (called "fail-safe mode on"), as well as without any fault-tolerance mechanisms (called "fail-safe mode off"). Input data and performed computations in all three cases were identical. Test cases do not include forced failures—otherwise it would be impossible to compare fails-safe mode with regular MPI I/O and plain NVRAM cache.

Testbed environment
Tests were performed on two clusters: Lap06 (equipped with NVRAM simulation platform), and K2 (composed of about a hundred of nodes). Those different environments were used in order to provide more comprehensive results: hardware simulation platform in Lap06 is able to accurately reproduce expected NVRAM parameters, while the bigger K2 allows to show scalability. Table 2 provides detailed specification of clusters and nodes. Since NVRAM is slower than regular RAM, its simulation is based on the idea of adding additional delays for read and write operations on certain ranges of RAM addresses. The operating system can see simulated NVRAM as an ext4 file system using Direct Access for files (DAX),[2] together with the ability to omit page cache mechanisms for memmap'd files. Results in this paper are obtained with MPICH 3.2, Orange-FS 2.8.7 as PFS and the following configuration of simulated NVRAM:

- RAM bandwidth limited four times, which limits it at the level of about 9.5 GB/s,
- additional read latency at the level of 600 ns,
- additional write latency at the level of 2000 ns.

Applications
Firstly, results were obtained with a random walk microbenchmark—an application created for the purpose of measurement of I/O performance. In contrary to popular I/O benchmarks, it also makes the CPU work under heavy load in order to simulate high performance computations. Tests with no computations were unrealistically optimistic in favor of the NVRAM cache. The algorithm operates as follows:

1. Read a single data chunk.
2. Perform some iterations of Collatz conjecture.
3. Write a single data chunk.
4. Choose a new location (random distance and direction) and return to the first step.

[2] https://www.kernel.org/doc/Documentation/filesystems/dax.txt.

Table 2 Testbed environment

	Lap06	K2
Number of computing nodes	6	96
Number of PFS nodes	1	3
CPU	2 × Intel® Xeon®E5-4620	2 × Intel® Xeon E5345
RAM (GB)	15	8
Network	40 Gb/s Infiniband	10 Gb/s Ethernet
Storage	SSD	HDD
NVRAM simulation	17 GB, hardware simulation	4 GB, tmpfs

State of the application contains the whole file the algorithm operates on. Secondly, we benchmarked a 2D map search application in which a potentially large map is searched for in parallel for objects by processes of an MPI application. The map is partitioned into distinct subdomains each of which is assigned to a distinct MPI process. Each of these searches for pixels meeting a certain criteria (color) and if found, searches its immediate surrounding for occurrence of a certain number of objects or until a predefined radius has been reached. This can be used for simulation of spreading of diseases in smart agriculture, in military applications searching for enemy vehicles in close proximity to important objects etc. The map, together with markers on selected locations, creates application state that could be recreated after a failure.

Fail-safe mode overhead

For test purposes, Fig. 3 presents a random walk microbenchmark parameterized with a 10 GB input file, 1 kB of single data chunk, 1.5 millions of Collatz conjecture iterations and various numbers of algorithm iterations. Results of 2D map search are shown in Fig. 4 and were obtained with different map sizes and a buffer of 512 B.

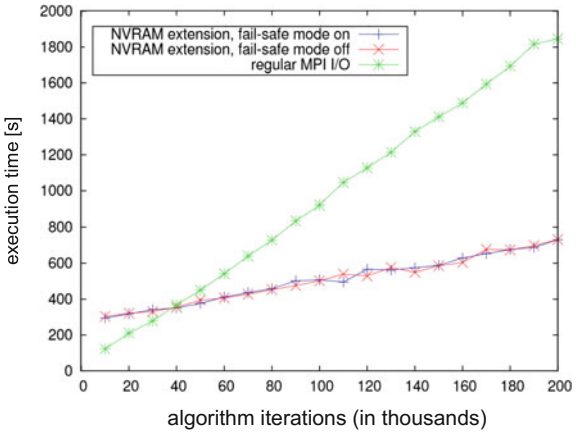


Fig. 3 Random walk microbenchmark, comparison of approaches

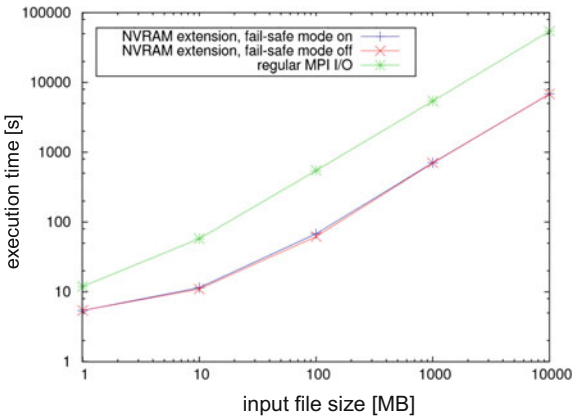


Fig. 4 2D map search, comparison between unmodified MPI IO and proposed extension

Both graphs prove, that the overhead introduced in the proposed fail-safe mode is negligible—a few microseconds per request/iteration per cache manager in the two applications. Moreover, the overhead does not depend neither on amount of access operations (represented by iterations of random walk algorithm), nor on the size of an input file (showed with different map sizes in Fig. 4). Results from this experiments also indicate that, if only an application meets certain criteria such as a high enough number of file operations per file open operation, unmodified MPI I/O implementation has a lower performance than proposed NVRAM cache based extension.

Scalability

Scalability of the solution was also tested with the random walk microbenchmark (10 GB of input file size, 1 kB of single data chunk, 13.5 millions of algorithm iterations, each iteration consisted of 1.5 millions of Collatz conjecture steps) and 2D map search (map of 10 GB). For both applications, the referential value of execution time running on a single process was obtained with a reduced file size, and then scaled according to computational complexity of the algorithm. This operation was required because a single node was not able to simulate enough NVRAM for storage of 10 GB file. The results presented in Fig. 5 show a nearly linear speedup of the random walk microbenchmark. Although connecting more nodes resulted in an increased intensity of I/O operations, each additional node participated in a distributed cache and the load was divided between more machines. Scalability of 2D map search was linear up to 168 processes, which is shown in Fig. 6. The speedup drop for higher numbers of processes is caused by too small an amount of computations—the implementation of searching through the map is not computationally demanding and after a certain number of processes is reached, the overhead for spawning and managing additional processes is higher that gain from parallel execution. What could be also noticed in this chart is that the solution performs better when data chunks are smaller.

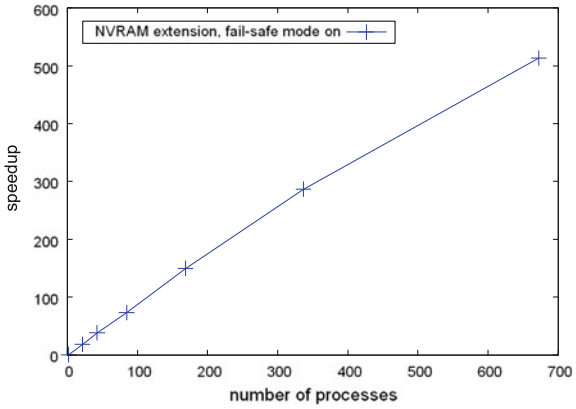


Fig. 5 Scalability illustrated with random walk microbenchmark on K2 cluster

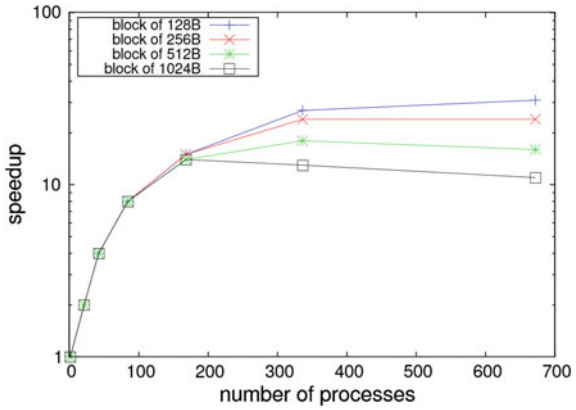


Fig. 6 Scalability illustrated 2D map search on K2 cluster

Recovery time

Recovery time results, measured on the Lap06 cluster, are shown in Table 3. When a simulated failure occurred, about 25 % of processes were sending 1 kB write requests. Recreation of a cache after a failure is a relatively cheap operation, because it does not require memory allocation or much data transmission. Moreover, execution time of this process does not depend on the size of a file.

Table 3 Recovery times for applications and various cache sizes

	Size of NVRAM cache		
	100 MB	1 GB	10 GB
Random walk microbenchmark (s)	9.899	10.016	9.922
2D map search (s)	10.001	9.809	9.926

5 Summary and Future Work

In this paper, we presented a solution, that allows to recover from failures caused by software, network and the environment. The solution is based on the MPI I/O extension supported by NVRAM cache. With persistence of NVRAM and the architecture of this extension, the overhead related to fail-safe mode is negligible, which was shown for two applications run in a real cluster environment. Experiments confirmed, that the extension is scalable, although it is strongly dependent on a specific application. We firmly believe, that the solution is an interesting alternative to checkpointing in environments, that do not need the highest level of fault tolerance.

In the future we would like to focus on testing the NVRAM based solution with a wider range of applications such as parallel graph search algorithms and geometric SPMD type applications. We also plan to further tune the method's initialization, experiment with replacement of threads with dynamically invoked MPI processes.

References

1. Rajachandrasekar, R., Moody, A., Mohror, K., Panda, D.: A 1 PB/s file system to checkpoint three million MPI tasks (2013)
2. Czarnul, P., Fraczak, M.: New user-guided and ckpt-based checkpointing libraries for parallel MPI applications. In: 12th European PVM/MPI Users' Group Meeting Sorrento, Italy. Springer LNCS 3666, pp. 351–358, 18–21 Sept 2005
3. Schroeder, B., Gibson, G.A.: Understanding failures in petascale computers. In: SciDac 2007: Scientific Discovery Through Advanced Computing. Volume 78 of Journal of Physics: Conference Series. Boston, MA, 24–28 June 2007
4. Smith, R.: Introducing Intel Optane Technology—Bringing 3D XPoint Memory to Storage and Memory Products (July 2015) https://newsroom.intel.com/presskits/introducing-intel-optane-technology-bringing-3d-xpoint-memory-to-storageand-memory-products/
5. Hargrove, P.H., Duell, J.C.: Berkeley lab checkpoint/restart (BLCR) for Linux clusters. In: SciDAC 2006. Volume 46 of Journal of Physics Conference Series, pp. 494–499 (2006)
6. Sankaran, S., Squyres, J., Barrett, B., Sahay, V., Lumsdaine, A., Duell, J., Hargrove, P., Roman, E.: The LAM/MPI checkpoint/restart framework: system initiated checkpointing. Int. J. High Perform. Comput. Appl. **19**(4), 479–493 (2005)
7. Dong, X., Muralimanohar, N., Jouppi, N., Kaufmann, R., Xie, Y.: Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future Exascale systems. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. SC'09, vol. 57, pp. 1–57:12. New York, ACM (2009)
8. Narayanan, D., Hodson, O.: Whole-system persistence with non-volatile memories. In: Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012), ACM (2012)
9. Gao, S., He, B., Xu, J.: Real-time in-memory checkpointing for future hybrid memory systems. In: Proceedings of the 29th ACM on International Conference on Supercomputing, pp. 263–272. ICS'15, New York, ACM (2015)
10. Dorozynski, P., Czarnul, P., Malinowski, A., Czurylo, K., Dorau, L., Maciejewski, M., Skowron, P.: Checkpointing of parallel MPI applications using MPI one-sided API with

support for byte-addressable non-volatile ram. ICCS 2016. Procedia Comput. Sci. **80**, 30–40 (2016)
11. Gropp, W., Lusk, E.: Fault tolerance in MPI programs. Spec. Issue J. High Perform. Comput. Appl. **18**, 363–372 (2002)
12. Fagg, G., Dongarra, J.: Building and using a fault-tolerant MPI implementation. Int. J. High Perform. Comput. Appl. **18**(3), 353–361 (2004)
13. Malinowski, A.., Czarnul, P.: Extension of MPI parallel i/o API (2015) https://github.com/pmem/mpi-pmem-ext/blob/master/mpiio_extension2/doc/mpiio_extension.pdf
14. Czarnul, P., Ciereszko, A., Fraczak, M.: Towards efficient parallel image processing on cluster grids using GIMP. In: Computational Science—ICCS 2004, Krakow, Poland. Lecture Notes in Computer Science 3037, pp. 451–458. Springer (2004)
15. Czarnul, P.: A model, design, and implementation of an efficient multithreaded workflow execution engine with data streaming, caching, and storage constraints. J. Supercomput. **63** (3), 919–945 (2013)

Towards Effective Allocation of Resources in Service-Oriented Systems

Łukasz Falas and Krzysztof Juszczyszyn

Abstract The article focuses on the problem of computing resource management in systems offering data processing methods in service-based model. It proposes a new model for service requests processing based on a concept of dynamic service selection in such a way, which meets client's requirements, while minimizing the service delivery cost for the service provider. The key innovation of the described approach is the utilization of machine learning methods for effective estimation of non-functional parameters and resource allocation decision-making.

Keywords Resource allocation · Service oriented architecture · Qos-aware service composition · Web services

1 Introduction

Dynamic development of Internet-based Web services had tremendous impact on service delivery strategies and, at the same time, influenced users' expectations which tend to be more focused on service quality. As for now, the base model of delivery is Software-as-a-Service (SaaS), which allows access from any device with Internet connection. Implementation of SaaS within the Service Oriented Architectures (SOA) greatly impacts the importance of service quality guarantees, service contracts and Service Level Agreements (SLAs). At the same time, details of the implementation are hidden from the users, who interact with services via standard, dedicated interfaces [4, 5]. In result, clients are unaware of implementation techniques, resource allocation schemes (CPU, memory, disk, and communication link usage).

Ł. Falas (✉) · K. Juszczyszyn
Chair of Computer Science, Wrocław University of Technology,
Wyb. Wyspiańskiego 27, 50-357 Wrocław, Poland
e-mail: lukasz.falas@pwr.wroc.pl

K. Juszczyszyn
e-mail: krzysztof.juszczyszyn@pwr.wroc.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_12

In this situation, a new challenges emerge in the context of resource allocation methods in SaaS systems. We expect that new types of service contracts, containing non-functional requirements will become common within the next few years. Service requests will be enriched with information about expected response time, service availability, security requirements etc. On the other hand, today's solutions based mainly on various strategies for task distribution between servers seem clearly inadequate to address these challenges and guarantee new types of SLAs. Recently released report "Computing Systems: Research Challenges Ahead: The HiPE-AC Vision 2011/2012" [10], together with numerous publications, indicate that key research areas connected with SOA and SaaS systems will be associated with quality and effectiveness guarantees. Within these areas, the management of computational and communication resources are being defined as key factors for the quality of Service guarantees [1, 17, 18]. The problem seems even more important in many real-world cases where services are responsible for computational tasks, intense data processing, decision support etc. In these cases, the actual values of non-functional parameters of services are directly dependent on resources allocated to services, and the size of data being processed. To make things more complicated, in most SOA and SaaS systems tasks may be executed by composite services, and there exist multiple instances of certain services which may differ in terms of allocated resources, response times etc. Each of these services may be chosen as a component of composite service which generates non-trivial optimization problems —allocation of resources to services which then take part of composite service execution plan should be done in a way guaranteeing the SLA.

This paper addresses the above challenges by proposing a new approach to the execution of composite services. Our approach assumes the application of several methods from machine learning and probabilistic models in order to estimate non-functional parameters of atomic and composite services, and to support optimal resource allocation. The preliminary experimental results will be also presented and discussed.

2 Related Work

Resource management in service-oriented systems was discussed in many works. But, in most cases, they were concentrated on fulfilling the functional requirements of the users (service composition), and non-functional requirements were addressed only in context of service provider's cost and the problem of the configuration of his computational resources. There are many papers covering the problem of composite service configuration, which means static binding of atomic services to certain composite services (as parts of their execution plan) [2, 3, 8, 14, 16]. The key problem with this approach is that it simplifies the dynamic nature of service

systems and does not address any type of resource reallocation (which may be sometimes inevitable). One of the first publications addressing dynamic service allocation during service request processing was [9], describing solutions implemented in eFlow system. This approach was, however, limited to the scenario of changing specific SLAs by the user. It was further developed within the e-health system based on composed Web services [13]. This work introduces the concept of service version, with service versions differing only in terms of non-functional parameters. Additionally, connection between resource allocation and service quality is discussed in this work.

In other work, the necessity of addressing changes in non-functional service parameters is discussed [19]. There is also a probabilistic model proposed for their estimation. Other probabilistic models were also proposed in [12, 15].

Our work stems from the solutions proposed in [6, 7] and offers their significant extension—a new model for service requests processing based on a concept of dynamic service selection, which meets client's requirements, while minimizing the service delivery cost for the service provider. The key component of the described approach is the utilization of machine learning methods for effective estimation of non-functional parameters and resource allocation decision-making.

3 Resource Allocation Task

Resource allocation task in service oriented systems is a problem which integrates access control, guarantees of non-functional service parameters, and service cost optimization (on service provider side). The pool of resources consists of various versions of services (with different non-functional characteristics), computational resources (processor time, memory, disk space), and communication resources (network links through which services communicate). Service requests, generated by users, come in a stream which is sequentially processed.

Dynamic service allocation task is executed when the request reaches the system, and involves the following stages:

- verification of availability of resources,
- computation of optimal resource allocation for composite service request
- decision about request acceptance or rejection (in case requirements cannot be met)
- allocation of resources and composite service execution.

Request response process in a dynamic resource allocation SaaS system is presented in Fig. 1.

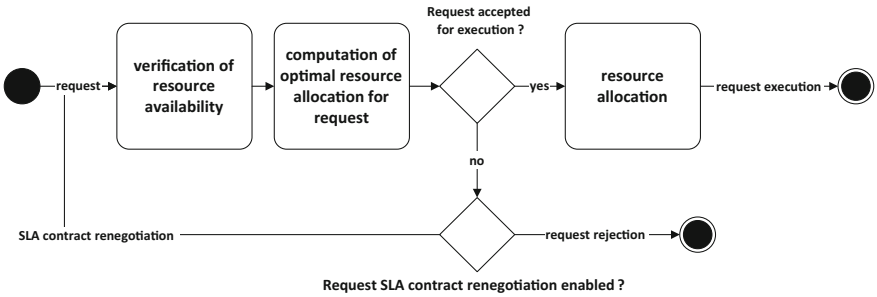


Fig. 1 Request processing in service-oriented system with optimization of resource allocation

4 Service System Model

In service-oriented systems, according to SOA reference architecture [1], we distinguish four layers: users, composite services, Web services, and infrastructure. In this architecture users generate requests and send them to the system, each of the composite services is composed of atomic services and has its own execution plan. We should note, that only atomic services are directly connected with infrastructure level—composite services are represented by their execution plans, which are processed by dedicated execution engines. Solution described in this paper is in fact a middleware placed between user and Web service layer. In order to solve the dynamic allocation of resources problem, we developed a mathematical model of service system, with the representation of non-functional parameters of services. It will be shortly described below.

Systems' resources are defined as a tuple:

$$SYSTEM = \{total\_cores, total\_memory\} \tag{1}$$

where total_cores is total number of computational cores in the system, while total_memory stands for memory resources. A request is defined as a tuple:

$$R_i = \{t_i, d_i, SLA_i\} \tag{2}$$

where i is the request number, t_i is request time, d_i is amount of data connected with the request, and SLA_i is a vector of non-functional requirements, defined as:

$$SLA_i = \left[nf_1, nf_2, \ldots, nf_j, \ldots, nf_J\right] \tag{3}$$

where nf_j stands for a constraint for jth non-functional parameter, and J is total number of non-functional parameters. The set of all servers available is defined as:

$$H = \{h_1, h_2, \ldots, h_k, \ldots, h_K\} \tag{4}$$

where k is the number of server, and K is the number of all servers available. The number of available cores on server k, (at the moment the request is submitted) which may be allocated to the request is a function:

$$core(k, t_i) \tag{5}$$

The amount of free memory on server k, (at the moment the request is submitted) which may be allocated to the request is a function:

$$memory(k, t_i) \tag{6}$$

A set of all atomic services in the system is defined as:

$$S = \{s_1, s_2, \ldots, s_l, \ldots, s_L\} \tag{7}$$

where l is the number of a service, and L is the number of all atomic services available. The set of all versions of an atomic service l is defined as:

$$SV_l = \{sv_{l1}, sv_{l2}, \ldots, sv_{lm}, \ldots, sv_{lM}\} \tag{8}$$

where m is the number of a service version, M jest the number of all services (including versions) available in the system, and sv_{lm} is a pair:

$$sv_{lm} = \{core_{lm}, memory_{lm}\} \tag{9}$$

where $core_{lm}$ is the number of cores used by version m of service l, and $memory_{lm}$ is memory used by version m of service l. Available link throughput between versions of services is defined as:

$$bandwidth\left(t_i, sv_{lmk}, sv_{opr}\right) \tag{10}$$

where sv_{lmk} is version m of service l executed on server k, and sv_{opr} is version p of service o executed on server r. A function (using regression method) which returns random variable SNP_{ijlm} describing the value if jth non-functional parameter for service version sv_{lm} at the time moment t_i is defined as:

$$SNP_{ijlm} = fs_{nf_j}(sv_{lm}, t_i)$$

Graph representing all possible execution plans of composite service executed in result of request R_i is defined as:

$$GS_i(\mathrm{VS_i} = \{\mathrm{V_{START}}, \mathrm{VC_1}, \mathrm{VC_2}, \ldots, \mathrm{VC_n}, \mathrm{V_{END}}\}, \mathrm{ES_i}) \tag{11}$$

where node VC_n is being mapped into set $\{sv_{n1}, sv_{n2}, \ldots, sv_{nm}\}$, which is a set of all versions of service n, which may be chosen to be the part of execution plan of composite service. Nodes $\mathrm{V_{START}}$ and $\mathrm{V_{END}}$ represent start and shutdown of composite service execution plan. $\mathrm{VS_i}$ is a set of all nodes of the composite service graph relevant to the request i, and $\mathrm{ES_i}$ is a set of all links in the graph, representing data flow between atomic services responding to request i as parts of executed composite service. Execution plan of composite service for request i is a defined as:

$$G_i(V_i, E_i) \tag{12}$$

where V_i represents all versions of services chosen to be executed as parts of the plan and E_i represents data flow between versions of services. Now we can define:

$$G_{if}\left(V_{if}, E_{if}\right) \in \mathrm{F}G_i \tag{13}$$

where $\mathrm{F}G_i$ is a set of all possible execution plans of composite service responding to request i (built from available services, on the basis of GS_i), and $G_{if}\left(V_{if}, E_{if}\right)$ is fth execution plan for a request i from the set $\mathrm{F}G_i$. For each non-functional requirement from SLA_i we can define a function aggregating the values of non-functional parameters of the fth execution plan for request i, which will return a random variable GNP_{ijf} describing the jth non-functional parameter:

$$GNP_{ijf} = fg_{nf_j}\left(G_{if}\left(V_{if}, E_{if}\right), \mathrm{t_i}\right) \tag{14}$$

The cost of service delivery may be defined as:

$$DC\left(G_{if}\left(V_{if}, E_{if}\right), \mathrm{SYSTEM}\right)$$

It returns a percentage value describing what part of total system's resources are resources allocated to the realization of request i by the execution plan of composite service f. Finally, basing on the above model we can define optimization criterion, allowing optimal allocation of resources within a composite service execution plan:

$$G_i(V_i, E_i) \leftarrow \min_{G_{if}\left(V_{if}, E_{if}\right)} DC\left(G_{if}\left(V_{if}, E_{if}\right), \mathrm{SYSTEM}\right)$$

Fulfilling constraints:

$$\begin{cases} fg_{nf_j}\left(G_{if}\left(V_{if}, E_{if}\right), t_i\right) \leq nf_j, & \text{for requirements with upper constraint} \\ fg_{nf_j}\left(G_{if}\left(V_{if}, E_{if}\right), t_i\right) \geq nf_j, & \text{for requirements with lower constraint} \end{cases}$$

The above model was used in the development of system allowing for optimal resource allocation and was implemented for the tests on real-life service repositories. Results are presented in the next section. Note, that this solution was developed for service systems using costly computational services for executing composite algorithms and decision making. It is also dedicated for systems applying the structural project pattern of the Façade [11], and introduces middleware responsible for processing of all queries and allocation of system's resources. It also uses the concept of dynamic resource allocation via virtualization on the operating system's level (like in Docker technology). In result, by default, no services are running when the request arrives, the virtualized container with a service is started on demand with time cost of 50–100 ms, with allocated resources determined by the machine learning algorithm.

5 Experiments

Experiments were focused on verification of machine learning methods applied to estimation of non-functional parameters of services taking into account the amount of data being sent, the number of processor cores being used and the allocated memory. Our method was compared with most prominent approaches found in literature (assuming that the average values of non-functional parameters are being used). The results shown below address primarily the most important non-functional parameter—the time. Two servers were used in the experiments, one in the role of management server (running our optimization middleware) and the second with Docker software which allows virtualization on operating system level. Each was equipped with 8 processor cores and 64 GB of RAM.

For the tests a special Web service was prepared for acquiring the current state of the processor cores and for precise allocation of them to specific services. The service was tested in three versions:

- Version I: 1 core and 1.5 GB of RAM
- Version II: 4 cores and 6 GB of RAM
- Version III: 8 cores and 12 GB of RAM

For experiments, a set of 200 requests was prepared, divided in proportions of 60:40 (learning set and test set for evaluating predictions). We have assumed, that for any request only one version of service is chosen, and then executed. The times of execution and processing of each request are recorded and stored. Our method

chooses a version of service, for which the estimated time of execution is being closely met by the user's requirements defined in the SLA. At the same time algorithm guarantees that this result is achieved with the minimum cost (amount of resources allocated to service).

Two runs were performed during the tests—one using standard methods of estimating non-functional parameters (average values), and the second using machine learning approach proposed here. The tests compared the two of the above approaches by using three measures (the results are presented below (Figs. 2 and 3):

- accuracy, defined as difference between estimated time of service execution and the actual time of execution,
- the number of requests which were served in accordance with the SLA,
- the amount of system's resource consumed.

The experiments proved that, in the case of data-processing services, the size of input data is an important factor, influencing estimation of execution time. As seen on the charts estimation based only on mean execution times (which neglects data size) results in significant inaccuracy. On the other hand, prediction of execution

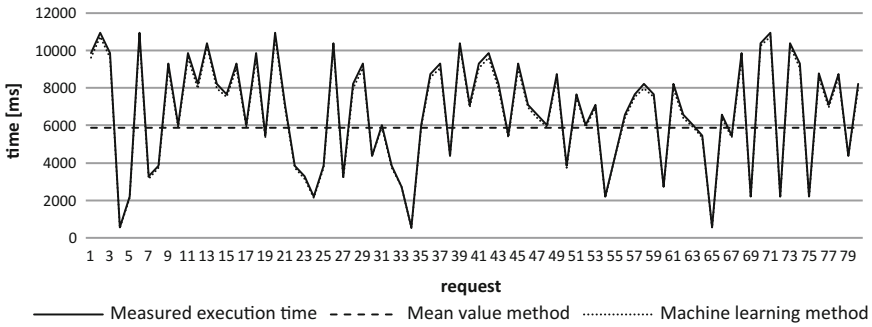


Fig. 2 Service execution time and time prediction for different methods (Version I)

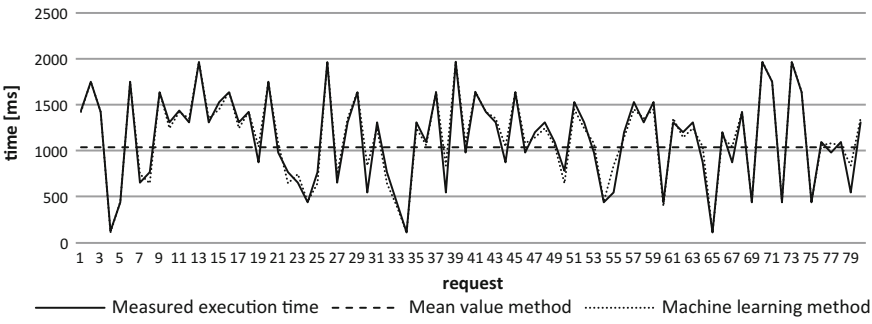


Fig. 3 Service execution time and time prediction for different methods (Version III)

time based on polynomial regression returns much better results. More detailed analysis of results is presented below. We can also note (Fig. 4) that in many cases time estimation based on mean values leads to the violation of SLAs (real execution time of services exceeds the value defined in SLA), while machine learning correctly estimates execution time and allows to choose appropriate versions of services (with more allocated resources) (Table 1).

The SLAs are not violated, which is a key feature of reliable and effective service system. Percentages of service requests processed in accordance with their SLAs are shown in the Table 2.

Figure 5 presents the utilization of processor cores by the services during the experiment, for both methods of service's execution time prediction methods. We can notice, that in some cases machine learning method allocates significantly more cores to the services, but, at the same time, no SLA is violated, while in the case of mean-based prediction the requirements of 45 % of SLAs were not met. Even more, machine learning method resulted in more effective resource allocation—in 50 % of cases it assigned less resources to the services than mean-based method (which was more effective only in 22.5 % of cases) (Table 3).

Summing up, our experiments have shown, that the machine learning method led to more effective allocation of system's resources, and significantly impacted service quality—no violations of SLAs were recorded. These results form the basis

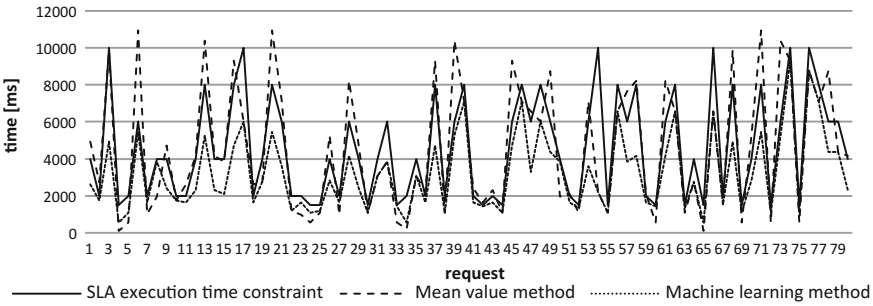


Fig. 4 Difference between measured and estimated service execution time

Table 1 Mean error of service execution time estimation

Service version	Mean value method	Machine learning method
I	2572.316	143.503
II	647.949	18.66
III	421.13	60.543

Table 2 Percentage of services processed in accordance with their SLAs

Method	Value (%)
Mean value method	55
Machine learning method	100

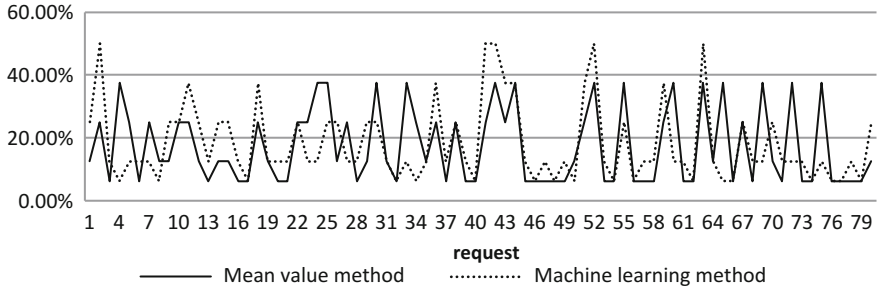


Fig. 5 Utilization of processor cores during experiment

Table 3 Utilization of system's resources

Measure	Value (%)
Mean value method—requests with better resource allocation	22.5
Machine learning method—requests with better resource allocation	50.0
Requests with the same resource allocation for both methods	27.5

for our next experiments, which will involve more composite services, and the development of dedicated algorithms and methods for the optimization of resource allocation in service systems.

6 Conclusions

This paper presents a new approach for processing requests in service-oriented systems. It was developed especially for systems, where services are responsible for composite computational and decision-supporting tasks, with processing of large volumes of data. Our approach implements the idea of dynamic allocation of resources at the moment of request submission and assumes virtualization (on the operating system level) with container technology (i.e. Docker). It may be also used in systems which utilize standard virtualization methods, but may be ineffective in such cases (starting virtual machine is time-costly). A new feature is the application of machine-learning methods for estimation of the values of non-functional parameters of Web services. Another important feature is allocation of resources to particular requests taking into account parameters like: computational resources, availability of communication links, the volume of data.

Our approach was tested on real-life examples of service repositories, and it was shown that it allows for better allocation of resources and minimizes the usage of computational infrastructure, at the same time guaranteeing the SLA agreements.

Further research will be concentrated on adaptation of the developed methods to more composite service requests and the development of dedicated algorithms for

faster decision making for resource allocation. The next experiments will be run using bigger service repositories and basing of the resources of recently set up Laboratory of Computing Clouds which is a part of national Polish PL-LAB2020 infrastructure.

References

1. Aazam, M., Khan, I., Alsaffar, A.A., Huh, E.-N.: Cloud of things: integrating internet of things and cloud computing and the issues involved. In: 11th International Bhurban Conference on Applied Sciences and Technology, pp. 414–419 (2014)
2. Bao, H., Dou, W.: A.: QoS-aware service selection method for cloud service composition. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 2254–2261 (2012)
3. Blanco, E., Cardinale, Y., Vidal, M., Graterol, J.: Techniques to produce optimal web service compositions. In: IEEE Congress on Services, pp. 553–558 (2008)
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Services Architecture, W3C Working Group Note, 11 Feb 2004 http://www.w3.org/TR/ws-arch, 2004
5. Brown, P.F., Metz, R., Hamilton, B.A.: Reference model for service oriented architecture 1.0, OASIS (2006)
6. Canfora, G., Di Penta, M., Esposito, R., Villani M.L.: A lightweight approach for QoS-aware service composition. In: 2nd International Conference on Service Oriented Computing ICSOC'04 (2004)
7. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: Qos-aware replanning of composite web services. In: IEEE International Conference on Web Services (ICWS'05), pp. 121–129 (2005)
8. Cao, H., Feng, X., Sun, Y., Zhang, Z., Wu, Q.: A service selection model with multiple QoS constraints on the MMKP. In: Network and Parallel Computing Workshops, pp. 584–589 (2007)
9. Casati, F., Shan, M.C.: Dynamic and adaptive composition of e-services". Inf. Syst. **26**(3), 143–163 (2001)
10. Duranton, M., Black-Schaffer, D., Yehia, S., de Bosschere, K.: Computing systems: research challenges ahead: the HiPEAC vision 2011/2012 (2011)
11. Erl, T.: Service Façade, SOA Design Patterns (1st edn.), pp. 333–343 (2009)
12. Falas, Ł., Stelmach, P.: Web Service composition with uncertain nonfunctional parameters. In: Technological Innovation for the Internet of Things, pp. 45–52 (2013)
13. Grzech, A., Świątek, P., Rygielski, P.: Dynamic resources allocation for delivery of personalized services. In: IFIP Advances in Information and Communication Technology: Software Services for e-World 341, pp. 17–28 (2011)
14. Huang, A.F.M., Lan, C., Yang, S.J.H.: An optimal QoS-based Web service selection scheme. Inf. Sci. **179**(19), 3309–3322 (2009)
15. Hwang, S.Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Inf. Sci. **177**, 5484–5503 (2007)
16. Juszczyszyn, K., Stelmach, P., Grzelak, T.: A method for the composition of semantically described Web services. In: Information Systems Architecture and Technology: Networks and Networks' services, pp. 27–37 (2010)
17. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Key challenges in cloud computing: enabling the future internet of services. IEEE Internet Comput. **17**(4), 18–25 (2013)
18. Sadiku, M.N.O., Musa, S.M., Momo, O.D.: Cloud computing: opportunities and challenges. IEEE Potentials **33**(1), 34–36 (2014)
19. Silva Cardoso, A.J.: Quality of Service and Semantic Composition of Workflows. PhD Dissertation, University of Georgia (2014)

Part IV
Communication Systems

Transient Processing Analysis in a Finite-Buffer Queueing Model with Setup Times

Wojciech M. Kempa and Dariusz Kurzyk

Abstract A finite-buffer queueing model with Poisson arrivals and generally distributed processing times is investigated. Every time when the service station restarts the operation after the idle period, a random-length setup time is needed to achieve full readiness for the work, during which the service process is suspended. A system of integral equations for time-dependent departure process, conditioned by the initial buffer state, is built. The solution of the corresponding system written for double transforms is obtained in a compact form. Hence the mean number of packets completely processed up to fixed time epoch can be easily found. The analytical approach is based on the idea of embedded Markov chain, total probability law and integral equations. The considered queueing system can be successfully used in cellular networks or WSNs modelling, where the setup time corresponds to leaving the sleep mode in energy saving mechanism. Numerical utility of analytical formulae is shown in a network-motivated computational example.

Keywords Departure process · Finite-buffer queue · Integral equation · Setup time · Transient state

1 Introduction

Evidently, queueing models with finite buffer capacities have wide network applications, especially in modelling some processes occurring in network nodes, like IP routers or nodes of WSNs. As it seems, systems with different-type limi-

W.M. Kempa (✉) · D. Kurzyk
Silesian University of Technology, Institute of Mathematics, ul. Kaszubska 23,
44-100 Gliwice, Poland
e-mail: wojciech.kempa@polsl.pl

D. Kurzyk
e-mail: dariusz.kurzyk@polsl.pl

W.M. Kempa · D. Kurzyk
Institute of Theoretical and Applied Informatics, Polish Academy of Sciences,
ul. Bałtycka 5, 44-100 Gliwice, Poland

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*,
Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_13

tations in the access to the service station are of particular importance, due to their potential adoptions as models of energy saving mechanisms. Typically, each busy period of the queueing system may correspond to the active mode and any idle period—to the sleep mode. In practice, when a packet (job, call, etc.) arrives into the empty system, it is impossible to start its operation immediately. Predominantly, it takes some time (called a setup time) for the server to achieve the full readiness for processing.

The steady-state behavior of the M/M/1-type queue with "queued" waking up and setup times is investigated in [2]. A Markovian system with server setups preceding the first service in a new busy period is analyzed in [13]. In [15] (see also [1]) an equilibrium threshold strategy for customers' behavior in a queue with setup times is derived. Applications of queueing systems with setup periods in WSNs can be found, e.g. in [16], where a sleep/wakeup protocol in the IEEE 802.15.4 standard is modelled. In [3] a queueing system in modelling the IMS session re-setup delay in WiMAX/LTE heterogeneous networks is applied. A BS sleeping mode in cellular networks is in [14] modelled via the M/G/1-type queueing system with server vacations and setup times. In [4] a model of data center with servers having independent setup times after idle periods is considered. The system is mathematically described by a two-dimensional Markov chain and some performance measures are found using a generating function approach.

As one can observe, analytical results obtained for different-type queueing models with server setup times relate, mainly, to the stable queues, i.e. to the stochastic characteristics in the case of $t \to \infty$. However, quite often time-dependent analysis of the system behavior seems to be more desired, in particular due to the high variability network traffic, e.g. in TCP/IP connections. Moreover, in rare traffic (like in some WSNs) the system stabilizes longer, so the investigation of its performance shortly after the opening or the application of a new control mechanism requires transient analysis (at fixed time t). Transient analysis of the queue-size distribution in the M/G/1-type queueing model with arrivals in random batches, N-policy and server setup times can be found in [7]. In [5] a similar model is considered with additional multiple vacation policy. Time-dependent solution for the queue-size distribution in a model with Poisson arrivals and setup/closedown times is obtained in [11].

In the paper we deal with the finite-buffer queueing model of the M/G/1 type in which the first processing in each new busy period (after finishing the idle time) is preceded by a generally-distributed setup time, during which the service process is still suspended (the arriving packets accumulate in the buffer queue). Using the idea of embedded Markov chain, a system of integral equations for the distribution of the number of packets completely processed up to the fixed time t (departure process), conditioned by the initial buffer state, is built. The solution of the corresponding system written for double transforms is obtained via linear algebraic approach and presented in a compact form by using a certain recursively defined sequence.

In [9, 10] new results for departure process in the model with "queued" waking up (N-policy) can be found. The same characteristic in the finite-buffer system with

auto-correlated input stream modelled by the BMAP process is studied in [8]. In [6] time-dependent results for departure process can be found for the model with multiple vacation policy.

2 Description of the Model and Auxiliary Results

In the paper we investigate a single-server finite-buffer queueing model in which packets arrive according to a Poisson process with rate λ and are being served individually with a general-type CDF (=cumulative distribution function) $F(\cdot)$ of the processing time, according to the FIFO service discipline. The maximal number of packets present in the system simultaneously equals K, i.e. we have $K-1$ places in the buffer queue and one place "in service station". Obviously, packets arriving during the buffer overflow period (when the server is occupied and the buffer is saturated) are being lost. We assume that the buffer may contain a number of packets being accumulated before the start of the system at time $t=0$. Each busy period, which is initialized together with the first arrival after the idle time, is preceded by a setup time, which is a random variable with a general-type CDF $G(\cdot)$. The server needs a setup time to achieve full operational readiness. If the system is empty before the opening, then it also starts a setup time at the moment of the first arrival. We assume independence of all inter-arrival, processing and setup times in the evolution of the system.

Let us denote by $h(t)$ the (random) number of packets completely processed up to the fixed time t, and define the distribution function of $h(t)$, conditioned by the initial level of buffer saturation, as follows:

$$H_n(t,m) \stackrel{\text{def}}{=} \boldsymbol{P}\{h(t)=m|X(0)=n\}, \tag{1}$$

where $t>0, m\geq 0, 0\leq n\leq K$ and $X(0)$ stands for the number of packets present in the system at the opening (at time $t=0$). The (conditional) departure process defined in (1) is one of the most important operating characteristics of each queueing system, illustrating its performance. Moreover, in network applications, the output stream of packets transmitted from one node of the network becomes the arriving stream of packets into another node.

We are interested in finding the explicit representation for the PGF (=probability generating function) of the LT (=Laplace transform) of $H_n(t,m)$, i.e. for the functional

$$\tilde{h}_n(s,z) \stackrel{\text{def}}{=} \sum_{m=0}^{\infty} z^m \int_0^{\infty} e^{-st} H_n(t,m)dt, \tag{2}$$

where $|z|<1$ and $\Re(s)>0$. In further analysis we use the following result from linear algebra which can be found in [12]:

Lemma 1 *Introduce two number sequences* $(\alpha_k), k \geq 0$, *and* $(\psi_k), k \geq 1$, *with the assumption* $\alpha_0 \neq 0$. *Each solution of the following system of linear equations with respect to* $x_n, n \geq 0$:

$$\sum_{k=-1}^{n} \alpha_{k+1} x_{n-k} - x_n = \psi_n, n \geq 0, \tag{3}$$

can be written in the form

$$x_n = CR_{n+1} + \sum_{k=0}^{n} R_{n-k}\psi_k, n \geq 0, \tag{4}$$

where C *is a constant independent on* n, *and* (R_k) *is the sequence (called a potential in* [12]) *connected with the* (α_k) *in the following way*:

$$\sum_{k=0}^{\infty} \theta^k R_k = \frac{1}{P_\alpha(\theta) - 1}, \tag{5}$$

where

$$P_\alpha(\theta) \stackrel{\text{def}}{=} \sum_{k=-1}^{\infty} \theta^k \alpha_{k+1}, |\theta| < 1. \tag{6}$$

Moreover, in [12] *it is proved that successive terms of the sequence* (R_n) *can be found recursively as follows*:

$$R_0 = 0, R_1 = \alpha_0^{-1}, R_{k+1} = R_1 \left(R_k - \sum_{i=0}^{k} \alpha_{i+1} R_{k-i} \right), k \geq 1. \tag{7}$$

In the further analysis we use the nomenclature $\bar{L}(x) \stackrel{\text{def}}{=} 1 - L(x)$, *where* $L(\cdot)$ *stands for arbitrary CDF, and the notation* $I\{A\}$ *for the indicator of the random event* A. *Moreover, let us define*

$$f(s) \stackrel{\text{def}}{=} \int_0^{\infty} e^{-st} dF(t), \tag{8}$$

$$g(s) \stackrel{\text{def}}{=} \int_0^{\infty} e^{-st} dG(t), \Re(s) > 0. \tag{9}$$

3 Integral Equations for Conditional Departure Process

In this section, by using the idea of embedded Markov chain and the continuous version of the formula of total probability, we find a system of integral equations for $H_n(t,m)$ $(t>0, m\geq 0, 0\leq n\leq K)$, defined in (1). Next, we build the corresponding system written for double transforms of conditional departure process, i.e. for functionals $\tilde{h}_n(s,z)$ $(0\leq n\leq K, \mathcal{R}(s)>0, |z|<1)$, given in (2).

Assume, firstly, that the system is empty before the opening, so its evolution begins with idle period and the setup time begins simultaneously with the arrival epoch of the first entering packet. We can, in fact, distinguish three mutually excluding random events:

1. the first packet occurs before the moment t and the setup time also completes before t (we denote this event by $E_1(t)$);
2. the first packet arrives before t but the setup time ends after t ($E_2(t)$);
3. the first arrival occurs after time t ($E_3(t)$).

Let us introduce the following additional notation:

$$H_0^{(i)}(t,m) \stackrel{\text{def}}{=} \boldsymbol{P}\{(h(t)=m) \cap E_i(t) | X(0)=0\}, \tag{10}$$

where $t>0, m\geq 0$ and $i=1,2,3$. Obviously

$$\boldsymbol{P}\{h(t)=m|X(0)=0\} = H_0(t,m) = \sum_{i=1}^{3} H_0^{(i)}(t,m) \tag{11}$$

and

$$\tilde{h}_0(s,z) = \sum_{i=1}^{3}\sum_{m=0}^{\infty} z^m \int_0^{\infty} e^{-st} H_0^{(i)}(t,m)dt. \tag{12}$$

According to the random event $E_1(t)$, we obtain the following representation:

$$\begin{aligned} H_0^{(1)}(t,m) = \int_{x=0}^{t} \lambda e^{-\lambda x} dx \int_{y=0}^{t-x} \Bigg[\sum_{i=0}^{K-2} \frac{(\lambda y)^i}{i!} e^{-\lambda y} H_{i+1}(t-x-y,m) \\ + H_K(t-x-y,m) \sum_{i=K-1}^{\infty} \frac{(\lambda y)^i}{i!} e^{-\lambda y} \Bigg] dG(y) \end{aligned} \tag{13}$$

Indeed, the first summand on the right side of (13) relates to the situation in which the buffer does not become saturated during the setup time, while the second one—to the case of the buffer overflow occurring during this time. Similarly, considering $E_2(t)$, we obtain

$$H_0^{(2)}(t,m) = I\{m=0\}\int_0^t \lambda e^{-\lambda x}\bar{G}(t-x)dx. \tag{14}$$

Evidently, if the setup time completes after t, the service at time t is still blocked, so the only possibility is $m=0$. Similar argumentation explains the last case, namely

$$H_0^{(3)}(t,m) = I\{m=0\}e^{-\lambda t}. \tag{15}$$

From (13)–(15) we get, referring to (11),

$$\begin{aligned} H_0(t,m) = &\int_{x=0}^{t} \lambda e^{-\lambda x}dx \int_{y=0}^{t-x}\left[\sum_{i=0}^{K-2}\frac{(\lambda y)^i}{i!}e^{-\lambda y}H_{i+1}(t-x-y,m)\right. \\ &\left.+H_K(t-x-y,m)\sum_{i=K-1}^{\infty}\frac{(\lambda y)^i}{i!}e^{-\lambda y}\right]dG(y) \\ &+I\{m=0\}\left(\int_0^t \lambda e^{-\lambda x}\bar{G}(t-x)dx+e^{-\lambda t}\right). \end{aligned} \tag{16}$$

Let us consider now the case of the system being non-empty at the opening (i.e. $1\le n\le K$). Since successive departure epochs are Markov (renewal) moments in the evolution of the M/G/1-type system, then, applying the continuous version of the formula of total probability with respect to the first departure epoch after $t=0$, we obtain the following system of integral equations:

$$\begin{aligned} H_n(t,m) = I\{m\ge 1\}&\int_0^t\left[\sum_{i=0}^{K-n-1}\frac{(\lambda x)^i}{i!}e^{-\lambda x}H_{n+i-1}(t-x,m-1)\right. \\ &\left.+H_{K-1}(t-x,m-1)\sum_{i=K-n}^{\infty}\frac{(\lambda x)^i}{i!}e^{-\lambda x}\right]dF(x)+I\{m=0\}\bar{F}(t), \end{aligned} \tag{17}$$

where $1\le n\le K$. Let us explain in a short form successive summands on the right side of (17). The first summand under the integral describes the situation in which there are some free places in the buffer before the first departure occurring at time $0<x<t$, while the second one corresponds to the case of the buffer saturation occurring before time t. In the last summand the first packet leaves the system after t. Introducing the double transforms $\tilde{h}_n(s,z)$, defined in (2), and utilizing the following identities:

$$\sum_{m=0}^{\infty} z^m \int_{t=0}^{\infty} e^{-st}dt \int_{x=0}^{t} \lambda e^{-\lambda x}dx \int_{y=0}^{t-x} \frac{(\lambda y)^i}{i!} e^{-\lambda y} H_j(t-x-y,m)dG(y) = a_i(s)\tilde{h}_j(s,z), \tag{18}$$

where we define

$$a_i(s) \stackrel{def}{=} \frac{\lambda}{\lambda+s} \int_0^{\infty} \frac{(\lambda y)^i}{i!} e^{-(\lambda+s)y} dG(y), \tag{19}$$

and

$$\sum_{m=0}^{\infty} z^m I\{m=0\} \int_{t=0}^{\infty} e^{-st} \left[\int_{x=0}^{t} \lambda e^{-\lambda x} \bar{G}(t-x)dx + e^{-\lambda t} \right] dt = \frac{\lambda[1-g(s)]+s}{s(\lambda+s)} \stackrel{\text{def}}{=} \beta(s,z) = \beta(s), \tag{20}$$

we transform the Eq. (16) to the following one:

$$\tilde{h}_0(s,z) = \sum_{i=0}^{K-2} a_i(s)\tilde{h}_{i+1}(s,z) + \tilde{h}_K(s,z) \sum_{i=K-1}^{\infty} a_i(s) + \beta(s). \tag{21}$$

Putting

$$\alpha_i(s,z) \stackrel{\text{def}}{=} z \int_0^{\infty} e^{-(\lambda+s)x} \frac{(\lambda x)^i}{i!} dF(x), \tag{22}$$

where $\Re(s) > 0$ and $|z| < 1$, we rewrite now (17) in the form

$$\tilde{h}_n(s,z) = \sum_{i=0}^{K-n-1} \alpha_i(s,z)\tilde{h}_{n+i-1}(s,z) + \tilde{h}_{K-1}(s,z) \sum_{i=K-n}^{\infty} \alpha_i(s,z) + \frac{1-f(s)}{s}, \tag{23}$$

where $1 \le n \le K$.

Let us apply to equations of the system (21) and (23) the following substitution:

$$\tilde{d}_n(s,z) \stackrel{\text{def}}{=} \tilde{h}_{K-n}(s,z), 0 \le n \le K. \tag{24}$$

Now we obtain from (23) the following equations:

$$\sum_{i=-1}^{n} \alpha_{i+1}(s,z)\tilde{d}_{n-i}(s,z) - \tilde{d}_n(s,z) = \phi_n(s,z), \tag{25}$$

where $0 \le n \le K-1$ and functionals $\phi_n(s,z)$ are defined as follows:

$$\phi_n(s,z) \stackrel{\text{def}}{=} \alpha_{n+1}(s,z)\tilde{d}_0(s,z) - \tilde{d}_1(s,z) \sum_{i=n+1}^{\infty} \alpha_i(s,z) - \frac{1-f(s)}{s}. \tag{26}$$

Similarly, introducing (24) into (21), leads to the following representation:

$$\tilde{d}_K(s,z) = \sum_{i=0}^{K-2} a_i(s)\tilde{d}_{K-i-1}(s,z) + \tilde{d}_0(s,z) \sum_{i=K-1}^{\infty} a_i(s) + \beta(s). \tag{27}$$

4 Compact-Form Solution for Transforms

In this section, utilizing Lemma 1, we find the solution of the system (25) and (27) in the compact form. Let us note that the system (25) has the same form as (3) with unknowns $\tilde{d}_n(s,z), n \ge 0$, but with functional coefficients, namely $\alpha_{i+1}(s,z)$ and $\phi_i(s,z)$, $i \ge 0$. Hence, the solution of (25) can be found by applying (4). Moreover, observe that the number of equations in (25) is finite comparing to (3). This fact can be used to find the value of $C = C(s,z)$ explicitly, treating the Eq. (27) (written for $n = K$) as a specific-type boundary condition.

Thus, for $0 \le n \le K$ the following formula holds true [compare (4)]:

$$\tilde{d}_n(s,z) = C(s,z)R_{n+1}(s,z) + \sum_{i=0}^{n} R_{n-i}(s,z)\phi_i(s,z), \tag{28}$$

where $n \ge 0$, and [see (7)]

$$\begin{aligned} R_0(\mathrm{s},\mathrm{z}) = 0, R_1(\mathrm{s},\mathrm{z}) = \alpha_0^{-1}(s,z), R_{k+1}(\mathrm{s},\mathrm{z}) \\ = R_1(s,z)\left[R_k(\mathrm{s},\mathrm{z}) - \sum_{i=0}^{k} \alpha_{i+1}(s,z)R_{k-i}(s,z)\right], \quad k \ge 1, \end{aligned} \tag{29}$$

and the functional sequence $\alpha_{i+1}(s,z)$ was defined in (22). Substituting $n = 0$ into (28), we get

$$\tilde{d}_0(s,z) = C(s,z)R_1(s,z). \tag{30}$$

Next, taking $n = 1$ in (28) and applying (26) and (30), we obtain

$$\begin{aligned}\tilde{d}_1(s,z) &= C(s,z)R_2(s,z) + R_1(s,z)\phi_0(s,z)\\ &= C(s,z)R_2(s,z) + R_1(s,z)\left[\alpha_1(s,z)R_1(s,z)C(s,z)\right.\\ &\quad\left.-\tilde{d}_1(s,z)\sum_{i=1}^{\infty}\alpha_i(s,z) - \frac{1-f(s)}{s}\right]\end{aligned} \tag{31}$$

and hence

$$\tilde{d}_1(s,z) = \theta(s,z)\left[C(s,z)\big(R_2(s,z) + \alpha_1(s,z)R_1^2(s,z)\big) - R_1(s,z)\frac{1-f(s)}{s}\right], \tag{32}$$

where

$$\theta(s,z) = \theta(s) \stackrel{\text{def}}{=} \left[1 + R_1(s,z)\sum_{i=1}^{\infty}\alpha_i(s,z)\right]^{-1} = \frac{f(\lambda+s)}{f(s)}. \tag{33}$$

From representations (30)–(33) follows that $\phi_i(s,z)$ [defined in (26)] for any fixed $i \geq 0$ can be written as a function of $C(s,z)$, which is the only unknown functional. To find the formula for $C(s,z)$, let us, firstly, write (27), applying on the right side of identities (26), (28), (30) and (32), and utilizing the fact that

$$\sum_{i=0}^{K-2} a_i(s)\tilde{d}_{K-i-1}(s,z) = \sum_{i=1}^{K-1} a_{K-i-1}(s)\tilde{d}_i(s,z). \tag{34}$$

We obtain

$$\begin{aligned}\tilde{d}_K(s,z) &= \sum_{i=1}^{K-1} a_{K-i-1}(s)\left[C(s,z)R_{i+1}(s,z) + \sum_{j=0}^{i} R_{i-j}(s,z)\phi_j(s,z)\right]\\ &\quad + C(s,z)R_1(s,z)\sum_{i=K-1}^{\infty} a_i(s) + \beta(s)\\ &= \sum_{i=1}^{K-1} a_{K-i-1}(s)\left[C(s,z)R_{i+1}(s,z)\right.\\ &\quad\left. + \sum_{j=0}^{i} R_{i-j}(s,z)\left(\alpha_{j+1}(s,z)\tilde{d}_0(s,z) - \tilde{d}_1(s,z)\sum_{r=j+1}^{\infty}\alpha_r(s,z) - \frac{1-f(s)}{s}\right)\right]\\ &\quad + C(s,z)R_1(s,z) = \Psi_1(s,z)C(s,z) + \chi_1(s,z),\end{aligned} \tag{35}$$

where we denote

$$\Psi_1(s,z) \stackrel{\text{def}}{=} \sum_{i=1}^{K-1} a_{K-i-1}(s)\left[R_{i+1}(s,z) + \sum_{j=0}^{i} R_{i-j}(s,z)\big(R_1(s,z)\alpha_{j+1}(s,z) - \theta(s)\big(R_2(s,z) + \alpha_1(s,z)R_1^2(s,z)\big)\sum_{r=j+1}^{\infty}\alpha_r(s,z)\right)\right] + R_1(s,z)\sum_{i=K-1}^{\infty} a_i(s) \tag{36}$$

and

$$\chi_1(s,z) \stackrel{\text{def}}{=} \sum_{i=1}^{K-1} a_{K-i-1}(s)\sum_{j=0}^{i} R_{i-j}(s,z)\left[R_1(s,z)\frac{1-f(s)}{s}\theta(s)\sum_{r=j+1}^{\infty}\alpha_r(s,z) - \frac{1-f(s)}{s}\right] + \beta(s). \tag{37}$$

Now let us substitute $n = K$ in (28) and utilize representations (26), (30) and (32). We obtain

$$\begin{aligned}\tilde{d}_K(s,z) &= C(s,z)R_{K+1}(s,z) \\ &\quad + \sum_{i=0}^{K} R_{K-i}(s,z)\left\{\alpha_{i+1}(s,z)R_1(s,z)C(s,z)\right. \\ &\quad \left. -\theta(s)\left[C(s,z)\big(R_2(s,z) + \alpha_1(s,z)R_1^2(s,z)\big) - R_1(s,z)\frac{1-f(s)}{s}\right]\sum_{j=i+1}^{\infty}\alpha_j(s,z) - \frac{1-f(s)}{s}\right\} \\ &= \Psi_2(s,z)C(s,z) + \chi_2(s,z),\end{aligned} \tag{38}$$

where

$$\Psi_2(s,z) \stackrel{\text{def}}{=} R_{K+1}(s,z) + \sum_{i=0}^{K} R_{K-i}(s,z)\left[\alpha_{i+1}(s,z)R_1(s,z) - \theta(s)\big(R_2(s,z) + \alpha_1(s,z)R_1^2(s,z)\big)\sum_{j=i+1}^{\infty}\alpha_j(s,z)\right] \tag{39}$$

and

$$\chi_2(s,z) \stackrel{\text{def}}{=} \sum_{i=0}^{K} R_{K-i}(s,z)\left(\theta(s)R_1(s,z)\frac{1-f(s)}{s}\sum_{j=i+1}^{\infty}\alpha_j(s,z) - \frac{1-f(s)}{s}\right). \tag{40}$$

From (35) and (38) follows immediately

$$C(s,z) = [\Psi_1(s,z) - \Psi_2(s,z)]^{-1}[\chi_2(s,z) - \chi_1(s,z)]. \tag{41}$$

Now the representations (24), (26), (28) and (41) lead to the following main theorem:

Theorem 1 *The representation for the PGF of the LT of the conditional departure process in the M/G/1/K-type model with generally distributed server setup times is following*:

$$\begin{aligned}\tilde{h}_n(s,z) &= \sum_{m=0}^{\infty} z^m \int_0^{\infty} e^{-st}\boldsymbol{P}\{h(t) = m | X(0) = n\}dt \\ &= [\Psi_1(s,z) - \Psi_2(s,z)]^{-1}[\chi_2(s,z) - \chi_1(s,z)]\Bigg\{R_{K-n+1}(s,z) \\ &\quad + \sum_{i=0}^{K-n} R_{K-n-i}(s,z)\left[\alpha_{i+1}(s,z)R_1(s,z) - \theta(s)\big(R_2(s,z) + \alpha_1(s,z)R_1^2(s,z)\big)\sum_{j=i+1}^{\infty}\alpha_j(s,z)\right]\Bigg\} \\ &\quad + \sum_{i=0}^{K-n} R_{K-n-i}(s,z)\left(\theta(s)R_1(s,z)\frac{1-f(s)}{s}\sum_{j=i+1}^{\infty}\alpha_j(s,z) - \frac{1-f(s)}{s}\right),\end{aligned} \tag{42}$$

where the formulae for $\alpha_i(s,z), R_i(s,z), \theta(s), \Psi_1(s,z), \chi_1(s,z), \Psi_2(s,z)$ and $\chi_1(s,z)$ are given in (22), (29), (33), (36), (37), (39) and (40), respectively.

Remark 1 Let us note that from (42) the formula for the conditional mean number $\boldsymbol{E}_n h(t)$ of packets completely processed until the fixed time epoch t (where $0 \le n \le K$) can be found. Namely, we have

$$\boldsymbol{E}_n h(t) = \frac{\partial}{\partial z}\left[\mathcal{L}^{-1}\big(\tilde{h}_n(s,z)\big)\right]_{z=1}, \tag{43}$$

where the notation $\mathcal{L}^{-1}[\cdot]$ stands for the inverse Laplace transform.

5 Numerical Results

Let us consider the stream of packets of average sizes 100 B, arriving into the node of a wireless sensor network according to a Poisson process. Consider three different arrival intensities 300, 400 and 500 Kb/s, which give $\lambda = 375$, $\lambda = 500$ and $\lambda = 625$ packets per second, respectively. Moreover, let us assume that a radio transmitter/receiver of the node is switched off during an idle period and needs an exponentially distributed setup time with mean 4 ms to become ready for processing. Besides, let packets are being transmitted with speed 500 Kb/s according

to 2-Erlang service distribution, that gives the mean processing time 1.6 ms (hence the parameter of the 2-Erlang service distribution is $\mu = 1250$). Under the assumptions about arrival and serving rates, the utilization factor ρ of the system equals to 0.6, 0.8 and 1.0, respectively. Mean number $\boldsymbol{E}_n h(t)$ of packets completely processed until the fixed time epoch t can be found from (43). Moreover, we can estimate the transient loss ratio function as $LR_n(t) \approx 1 - \frac{\boldsymbol{E}_n h(t)}{\lambda t}$, where λt is mean number of packets in the arrival stream up to the time t. Transient evolutions of the mean number $\boldsymbol{E}_0 h(t)$ of completely processed packets and the estimations of the loss ratio $LR_0(t)$ for given values of system parameters are presented in Figs. 1 and 2, respectively.

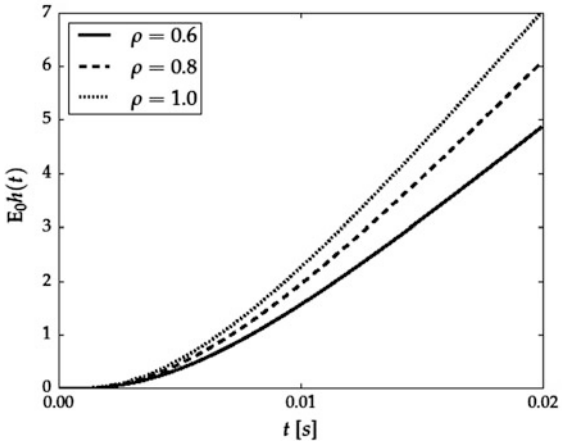


Fig. 1 Transient mean number $\boldsymbol{E}_0 h(t)$ of completely processed packets for ρ = 0.6, 0.8 and 1.0

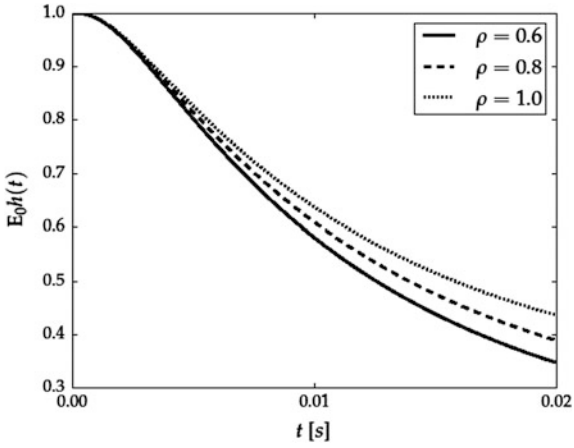


Fig. 2 Estimation of transient loss ratio $LR_0(t)$ for ρ = 0.6, 0.8 and 1.0

References

1. Burnetas, A., Economou, A.: Equilibrium customer strategies in a single server Markovian queue with setup times. Queueing Syst. **56**(3–4), 213–228 (2007)
2. Chen, P.S., Zhou, W.H., Zhou, J.W.: Equilibrium customer strategies in the queue with threshold policy and setup times. In: Mathematical Problems in Engineering. Optimization Theory, Methods, and Applications in Engineering, Hindawi Publishing Corporation (2015)
3. Edward, E.P.: A novel seamless handover scheme for WiMAX/LTE heterogeneous networks. Arab. J. Sci. Eng. **41**(3), 1129–1143 (2016)
4. Hu, J.N., Tuan, P.D.: Power consumption analysis for data centers with independent setup times and threshold controls. In: AIP Conference Proceedings, vol. 1648 (2015)
5. Kempa, W.M.: The transient analysis of the queue-length distribution in the batch arrival system with N-policy, multiple vacations and setup times. In: AIP Conference Proceedings, vol. 1293 (2010), 235–242 (Proceedings of 36th International Conference Applications of Mathematics in Engineering and Economics (AMEE'10), Sozopol, Bulgaria, 2010)
6. Kempa, W.M.: Analysis of departure process in batch arrival queue with multiple vacations and exhaustive service. Commun. Stat. Theory Methods **40**(16), 2856–2865 (2011)
7. Kempa, W.M.: On transient queue-size distribution in the batch arrival system with the N-policy and setup times. Math. Commun. **17**(1), 285–302 (2012)
8. Kempa, W.M.: Study on time-dependent departure process in a finite-buffer queueing model with BMAP-type input stream. In: Proceedings of the IEEE 2nd International Conference on Cybernetics (CYBCONF 2015), Gdynia, Poland (2015)
9. Kempa, W.M.: Time-dependent analysis of transmission process in a wireless sensor network with energy saving mechanism based on threshold waking up. In: Proceedings of the IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2015), Stockholm, Sweden (2015)
10. Kempa, W.M., Kurzyk, D.: Transient departure process in M/G/1/K-type queue with threshold server's waking up. In: Proceedings of the 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2015), Split—Bol (Island of Brac), Croatia, pp. 32–36 (2015)
11. Kempa, W.M., Paprocka, I.: Analytical solution for time-dependent queue-size behavior in the manufacturing line with finite buffer capacity and machine setup and closedown times. In: Applied Mechanics and Materials, vol. 809–910, 2015, pp. 1360–1365 (Selected, peer reviewed papers from the 19th Conference on Innovative Manufacturing Engineering (IManE 2015), Iasi, Romania, 2015)
12. Korolyuk, V.S.: Boundary-Value Problems for Compound Poisson Processes. Naukova Dumka, Kiev (1975)
13. Ma, Q.: Analysis of a clearing queueing system with setup times. RAIRO—Oper. Res. **49**(1), 67–76 (2015)
14. Niu, Z.S., Guo, X.Y., Zhou, S., Kumar, P.R.: Characterizing energy-delay tradeoff in hyper-cellular networks with base station sleeping control. IEEE J. Sel. Areas Commun. **33** (4), 641–650 (2015)
15. Sun, W., Guo, P.F., Tian, N.S.: Equilibrium threshold strategies in observable queueing systems with setup/closedown times. Central Eur. J. Oper. Res. **18**(3), 241–268 (2010)
16. Yue, W.Y., Sun, Q.T., Jin, S.F.: Performance analysis of sensor nodes in a WSN with sleep/wakeup protocol. Lect. Notes Oper. Res. **12**, 370–377 (2010)

Analysis of Routing Protocols Metrics for Wireless Mesh Networks

Piotr Owczarek, Maciej Piechowiak and Piotr Zwierzykowski

Abstract The article presents the survey of routing protocols for wireless mesh networks and their efficiency including the impact of routing metrics. Routing protocols are crucial element for performance of mesh networks, they ensure even flow of data packets and reduce negative influence of interferences. The article contains an overview of routing protocols used in the network mesh. As part of the research includes a comparative analysis of representative routing protocols carried out in the OMNeT++ environment. Results of simulation has been presented and compared.

Keywords Wireless mesh networks · Routing protocols · OMNeT++

1 Introduction

The popularity of wireless mesh networks is rapidly growing as it can be an answer for the last mile problem, a solution for rugged terrains or developing regions and countries. Wireless mesh networks are a promising technology, which is used for services such as client access to broadband Internet, monitoring systems, agglomeration, handling mass events and provide communications on the battlefield [1, 2]. This also allow to connect customers, which are not able to communicate with access device directly (non line of sight, NLOS).

Wireless mesh networks can reduce the transmit power through the use of directional antennas and accurate steering of the beam. This enables energy-efficient

M. Piechowiak
Kazimierz Wielki University, Bydgoszcz, Poland
e-mail: mpiech@ukw.edu.pl

P. Owczarek (✉) · P. Zwierzykowski
Poznan University of Technology, Poznan, Poland
e-mail: piotr.owczarek@verbicom.pl

P. Zwierzykowski
e-mail: piotr.zwierzykowski@put.poznan.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_14

equipment and reducing the coefficient of inter-channel interference. It is also possible to increase the transmission capacity and the rate of re-use of frequencies. By redundancy of transmission paths wireless mesh networks are characterized by high reliability, which is due to the possibility of quick reconfiguration in case of failure.

Unlike traditional wireless networks, based on a small number of access points connected by a cable infrastructure, mesh networks are composed of multiple nodes exchanging information with each other mainly by radio. The network nodes are acting as client nodes and routing nodes that direct packets to destinations and are responsible for maintaining and managing communication with neighbors.

Mesh networks also allow for integration with other networks (including the Internet) using gateways. Despite the above advantages wireless mesh networks still need to be refined in terms of MAC layer scalability and routing protocols used for routing packets between nodes. An important problem is the attack protection for protocols belonging to the four lowest layers of the reference model [3].

Mesh network routers have typically multiple network interfaces that allow for communication with a number of neighboring nodes. At the lower level of the transmitted signal power mesh network nodes provide the same coverage as infrastructure type known from WLAN topology. Client nodes usually have only one interface without built-in router functionality, which considerably simplifies the construction of the client platform [3].

Mesh networks are dynamically self-organizing, so that reduced the need for a central point of network management. Customers equipped with access radio card have the ability to connect directly to the mesh network, while others can take advantage of the bridges for the integration of both individual clients as well as entire networks operating in other technology than the mesh network, e.g. Ethernet, WiMAX and UMTS [3].

The article focuses on the efficiency of routing protocols in wireless mesh networks. The authors have selected the following set of parameters as the measure of the efficiency: packet loss ratio, packet delivery delay, average energy consumption of all network nodes and number of collisions in network. The article consists of six sections. In Sect. 2 representative routing protocols for wireless mesh networks are presented. Sect. 3 presents routing metrics while Sect. 4 describes the methodology used in the efficiency analysis presented in Sect. 5. Finally, Sect. 6 concludes the article.

2 Routing Protocols

Routing protocols use algorithms and optimization techniques forming the optimal path through the network [4–6] or minimum distribution tree [7–9]. Routing protocols for wireless mesh networks can be divided into two main types: proactive and reactive. Proactive protocols continuously maintain information in the routing tables about available routes by which the network node is achieved. It causes the

necessity of a continuous information exchange, updating the status of routing tables, which introduces additional fixed network traffic associated only with the process of routing. However, when there is a need for packet transmission, this type of protocols do not introduce delay needed to select the optimal transmission path.

Reactive protocols operate on demand and introduce additional delay required to locate the destination network node with the most effective path to the desired point of the network. The advantage of this approach is the lack of additional permanent datagram transmission delay, which is important especially for time-sensitive services such as speech transmission.

One of the most popular protocol used for both reactive MANET networks and wireless mesh networks is AODV protocol [10]. The AODV protocol is based on the DSDV protocol [11] and similarly uses the sequence numbers to determine the optimal path of the package. During the determination of the path to the destination node it checks the current routing table. If the address of the target device is not in routing table, RREQ message is sent to all neighbors. This procedure continues until the destination node is reached. Each router stores information about the sender of the message and ignore it in case of the copy of the same information. The reply to the source router is passed in a RREP message along the return path.

The AODV protocol manages the entries in the routing tables using HELLO messages, and when it detects a link failure it generates *Unsolicited Route Reply* message. AODV protocol does not require a broadcast transmission, which is its unquestionable advantage compared to DSDV. It also minimizes the number of nodes that do not participate in the transmission, contributing to the reduction of energy consumption.

The DYMO protocol [12] is based on two basic operations: route discovery and route maintenance. The first of these procedures are initiated when a source node wants to send a packet to the target device, which does not appear in the routing table. For this purpose, the route request message is broadcast on the network. When the destination node receives an addressed message—it transmits feedback information containing the cumulative path. Each intermediate node stores in its routing table the following information: the address of the destination node, the sequence number, the number of hops, next hop address, the name of the interface on the next hop, gateway and storage time of path in the table. The route maintenance is triggered when there is a change in the network topology [13].

The OLSR [14] protocol uses the concept of multipoint relays (MPs), which limit the network traffic and the number of link state updates. Each node operates in a minimum group of MPRs spaced by one hop. The set of MPR is formed by a rule providing availability of each of the nodes spaced apart by two hops. The information necessary for the operation of the network are exchanged only by the nodes that are in the group of MPR—other nodes do not broadcast protocol messages. HELLO messages sent periodically between routers are used to obtain information about nodes spaced about two hops. All devices in the network are periodically informed of the MPR group in *Communications Topology Control* messages. One of the features of the OLSR protocol is the redundancy of routes.

A key feature of the B.A.T.M.A.N. protocol [15] is a decentralized management system of routing path. Namely, none of the nodes do not have any information about the entire path the package travels. This eliminates the need to disseminate information concerning changes in the network to all nodes that are involved in communication. A single router stores only the information about the interface, the packet will be sent to the recipient. In this way, the routing process is run dynamically until it is received at the destination. This protocol also detects new network nodes and informs the neighboring routers. The quality of links is determined on the basis of *Transmit Link Quality* and *Receive Quality* factors for particular channels (used further in the deciding process of the optimal path routing).

The HWMP hybrid protocol is described in the 802.11 s standard for wireless mesh networks. It is a combination of a proactive routing and pro-active tree method. It uses a modified AODV protocol, which defines the metric based on the link parameters of the physical layer. The pro-active tree method builds a spanning tree from the root, which is usually a gateway while AODV protocol finds shortest path between nodes (HWMP supports two kinds of path selection protocols) [16].

3 Routing Metrics

A key element of each routing protocol are the metrics that enable the determination of the optimal transmission path. Metrics used in wireless mesh networks can be classified according to several criteria, which are widely discussed in the literature [17, 18]. The metrics that have been proposed for mesh networks can be divided as follows [19]:

- metrics related to the number of hops (*hop count*),
- metrics that determine the quality of a connection (*link quality*),
- metrics that are based on network load rate (*load-dependant* metrics),
- multi channel metrics.

Due to the simplicity and obsolescence of hop-count metric more attention should be given then to the remaining metrics. One can distinguish seven metrics based on the link quality [20]: *Expected Transmission Count* (ETX) [21], *Minimum Loss* (ML) [22], *Expected Transmission Time* (ETT) [21], *Expected Link Performance* (ELP) [21], *Per-Hop Round Trip Time* (RTT), *Per-Hop Packet Pair Delay* (PPD) and *Expected Transmission on a Path* (ETOP). Load-Dependent Metrics include: *Distribution Based Expected Transmission Count* (DBEXT) and *Bottleneck Aware Routing Metric* (BATD). The following multi-channel metrics stand out among other multi-channel metrics: *Weighted Cumulative ETT* (WCETT), *Metric of Interference and Channel-switching* (MIC) [17], *Modified ETX* (mETX) [23], *Effective Number of Transmissions* (ENT) [23], *iAWARE* [24] and *Exclusive Expected Transmission Time* (EETT) [25]. The comparison of the selected characteristics of metrics used in the majority of routing protocols for wireless mesh

networks is presented in [26]. The results were obtained by the authors from literature studies and the characteristics given on [19, 27].

4 Simulation Study

In order to conduct a comparative analysis, the authors selected simulation environment OMNeT++, which offers a wide range of configurations and implements the components of each layer of the reference model [28]. The graphical interface allows to create transparent network configuration. The functionality of wireless mesh networks and propagation conditions approximating actual transmission conditions have been implemented in the OMNeT++ in the form of INET and INETMANET libraries [29].

Library INETMANET includes protocols and components useful for modeling wireless network transmission, such as:

- propagation models,
- mobility models,
- data link layer standards 802.11,802.15.4, 802.16e,
- support of multiple radio interfaces,
- modeling operation on battery power,
- routing protocols specific to wireless networks (i.e. OLSR or AODV).

Network topology used in the study was developed on a rectangular plane with dimensions 2000 × 2000 m. It consists of 16 stationary nodes forming a backbone network used to route packets sent over a wireless hosts located on opposite sides of the rectangle. Each of the nodes has a limited transmit power, so it can only communicate with neighboring devices with one of the three available radio channels in a band of 5.2 GHz, according to the IEEE 802.11a. This reduces the power consumption, the level of interference between devices and expands the available spectrum resources. Each node has a radio channel data rate of 54 Mbps. Similar relation between parameters used for efficiency evaluation may be observed for different network topologies and data rates of radio channel [30].

5 Simulation Results

Figure 1 shows the average ratio of packet loss for all devices on the network depending on the routing protocol for wireless mesh network. Among the considered protocols the smallest average value (1.8 %) was obtained for OLSR protocol, because of its proactive way of functioning (it prevents from packet loss). At the opposite extreme are reactive protocols AODV, DYMO and HWMP with the loss rate of 2.7–3.7 % of all sent packets. It also confirms that the reactive routing protocols should be used in the case when the reliability of delivery in the network is required.

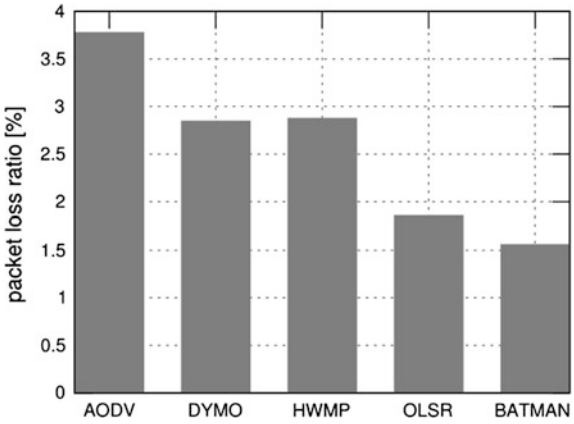


Fig. 1 Packet loss ratio for the selected routing protocols

Figure 2 shows the average packet delivery delay in the WMN networks. The highest value of delay is typical for reactive AODV protocol, for which the average delay is about 1.5 ms. The result of the simulation shows that DYMO and HWMP protocols are characterized by a delay of more than three times smaller than in AODV protocol and is approximately 0.4 ms. The value obtained for the OLSR protocol is 0.7 ms and is higher than for the DYMO and HWMP protocols. The expected delay value should be less than those obtained for the two reactive routing protocols, due to the fact that the protocol has a proactive knowledge of the routing path and does not require additional time for setting routes.

Average energy consumption for all devices on the network for the selected routing protocols is presented on Fig. 3. The values in the histogram represent the total value of the units of simulation in which the radio interfaces in the network have been in use. In the case of the AODV protocol the resulting value is highest.

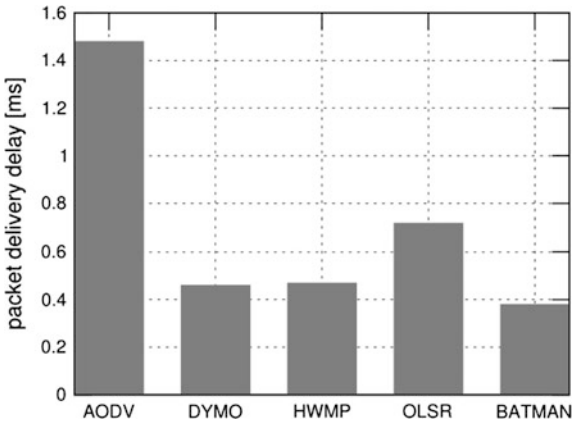


Fig. 2 Packet delivery delay for the selected routing protocols

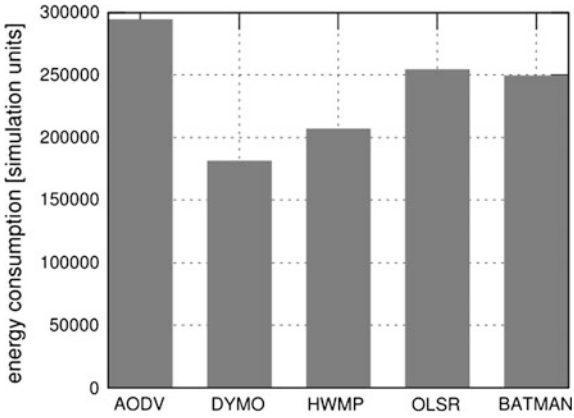


Fig. 3 Average energy consumption of network nodes for the selected routing protocols

The usage of other protocols resulting in lower energy consumption. The DYMO protocol (forming the simplification of AODV protocol) results are reduced by one third. For other protocols the energy consumption, with respect to the AODV, is reduced by 30 % for HWMP and 20 % for OLSR. The usage of the proactive routing protocols require continuous exchange of information between network nodes, which translates into a constant energy consumption, even when the transmission of user data is not carried out. The obtained results show that although the AODV protocol belongs to a group of reactive protocols, the energy consumption on the network is greater than the energy consumption of OLSR proactive protocol.

Figure 4 shows the total number of collisions on the network for each routing protocol used in the wireless mesh network. The number of collisions illustrates the effectiveness of transmission path differentiation. The best results were obtained for DYMO and OLSR protocols, for which the total number of collisions in the network does not exceed the value of 2200. The AODV protocol causes twice as many

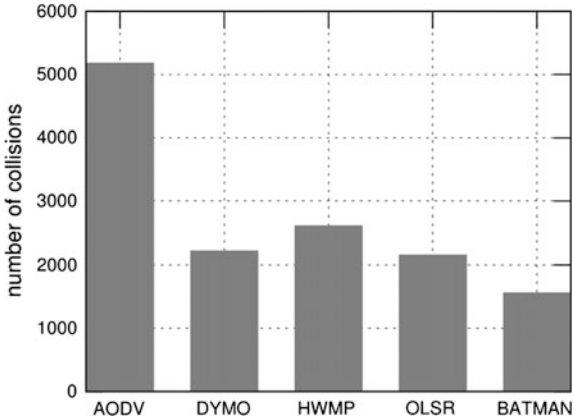


Fig. 4 Number of collisions occurs in network for the selected routing protocols

Table 1 The impact of the type of metrics on the basic transmission parameters of OLSR routing protocol

Transmission parameter	OLSR protocol with hop-count	OLSR protocol with ELX
Packet delivery delay (ms)	0.78	0.54
Packet loss ratio (%)	1.86	0.9
Number of collisions	2163	1352
Volume of network traffic (kB)	331	361

collisions than other considered protocols. The reason for generating more collisions than other protocols is the reactive mode, which calculates routes on demand. On the other hand, the AODV protocol uses a metric based on the number of hops, which means that it always selects the same shortest path. This translates directly to the number of collisions, due to the limited resources of time and spectral.

In order to determine the value of a metric, the network node measures the number of lost packets on the downlink and uplink and calculates the probabilities in the respective directions. The metric of the entire path is calculated as the sum of the ETX of all links involved in packet transmission from the sender to the recipient and the path with the minimal total value of ETX is chosen. Calculations are performed assuming the independence of the packet loss rate of its size.

Table 1 shows the comparative results of OLSR protocol with two types of metrics. The first of these metrics counts the number of hops and calculates the shortest path then. The second metric (ETX) bases on the number of lost packets. The following parameters were taken into account: packet delivery delay, packet loss ratio, number of collisions and volume of network traffic.

The implementation of ELX metric reduces the average value of the packet delivery delay and packet loss ratio decreases twice. The total amount of traffic is greater in the case of ETX metric. It is caused by the transmission of additional packets used for approximating the value of ETX metric.

6 Conclusions

In this article we studied the efficiency of network routing protocols for wireless mesh networks with OMNeT++. For this purpose, a unified model of the network configuration for both, reactive and proactive protocols was create. This will help to develop a consistent methodology of these protocols. The only variable in the simulation process was the routing protocol. The results indicate a lower latency in packet delivery and lower values of packet loss factor using proactive protocols. This is done by higher overhead of data sent over the network and higher energy consumption. However, due to the relatively wide bandwidth and lack of restrictions on battery power as is the case with network ad hoc networks, mentioned type of routing can be implemented in wireless mesh networks.

References

1. Kiedrowski, P.: Toward more efficient and more secure last mile smart metering and smart lighting communication systems with the use of PLC/RF hybrid technology. Int. J. Distrib. Sens. Netw. **2015**, 1–9 (2015)
2. Kiedrowski, P., Dubalski, B., Marciniak, T., Riaz, T., Gutierrez, J.: Energy greedy protocol suite for smart grid communication systems based on short range devices. In: Choraś, R.S. (ed.) Image Processing and Communications Challenges 3. Advances in Intelligent and Soft Computing, vol. 102, pp. 493–502. Springer, Berlin (2011)
3. Akyildiz, I.F., Wang, X.: Wireless Mesh Networks. Wiley, London (2008)
4. Czerniak, J.M., Dobrosielski, W., Apiecionek, Ł., Ewald, D.: Representation of a trend in OFN during fuzzy observance of the water level from the crisis control center. In: Proceedings of the Federated Conference on Computer Science and Information System FedCSIS 2015, Łódź, Poland, 2015, Annals of Computer Science and Information Systems, vol. 5, pp. 443–447. doi:10.15439/2015F217
5. Prokopowicz, P.: Flexible and simple methods of calculations on fuzzy numbers with the ordered fuzzy numbers model. In: Rutkowski, L. et al. (eds) Artificial Intelligence and Soft Computing. Proceedings of ICAISC 2013, Zakopane, Poland, Part I. LNAI, vol. 7894, pp. 365–375. Springer, Berlin (2013)
6. Głąbowski, M., Musznicki, B., Nowak, P., Zwierzykowski, P.: An algorithm for finding shortest path tree using ant colony optimization metaheuristic. In: Choraś, R.S. (Ed.) Image Processing and Communications Challenges 5, Advances in Intelligent Systems and Computing, vol. 233, pp. 317–326 (2014)
7. Bartczak, T., Zwierzykowski, P.: Lightweight PIM—a new multicast routing protocol. Int. J. Commun. Syst. **27**(10), 1441–1458 (2014)
8. Piechowiak, M., Zwierzykowski, P., Stachowiak, K., Bartczak, T.: Quality of multicast trees in ad-hoc networks with topology control. In: 9th International Symposium on Communication Systems, Networks & Digital Signal Processing, CSNDSP 2014, Manchester, UK, July 23–25, pp. 7–11 (2014)
9. Piechowiak, M., Zwierzykowski, P.: How to simulate and evaluate multicast routing algorithms. In: Pathan, A.K., Monowar, M.M., Khan S. (eds.) Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test, pp. 229–264. CRC Press (2015)
10. Perkins, C., Belding-Royer, E., Das, S.: Ad Hoc On-Demand Distance Vector (AODV) routing. IETF. RFC 3561 (2003)
11. Perkins, C., Bhagwat, P.: Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In: SIGCOMM'94 Proceedings of the conference on communications architectures, protocols and applications, pp. 234–244 (1994)
12. Chakeres, I.D., Perkins, C. E.: Dynamic MANET on demand (DYMO) routing protocol. Internet-Draft Version 06, IETF (2006)
13. Bisoyi, S.K., Sahu, S.: Performance Analysis of Dynamic MANET On-demand (DYMO) Routing protocol. Special Issue of IJCCT, 1.2 (2010):3 (2010)
14. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC 3626 (2003)
15. Neumann, A., Aichele, C., Lindner, M., Wunderlich, S.: Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). Internet-Draft (2008)
16. Bari, S.M.S., Anwar, F., Masud, M.H.: Performance study of hybrid wireless mesh protocol (HWMP) for IEEE 802.11s WLAN Mesh Networks. In: 2012 International Conference on Computer and Communication Engineering (ICCCE), IEEE (2012)
17. Yang, Y., Wang, J., Kravets, R.: Designing routing metrics for MESH networks. IEEE Workshop on Wireless MESH Networks (WiMESH) (2005)
18. Entezami, F., Politis, C.: Routing protocol metrics for wireless MESH networks. Wireless World Research Forum (2013)

19. Mogaibel, H.A., Othman, M.: Review of routing protocols and it's metrics for wireless mesh networks. In: Conference on Computer Science and Information Technology-Spring, pp. 62–70 (2009)
20. De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. Wireless Netw. **11**(4), 419–434 (2005)
21. Draves, R., Padhye, J., Zill, B.: Routing in multi-radio, multi-hop wireless mesh networks. In: Annual International Conference on Mobile Computing and Networking, pp. 114–128 (2004)
22. Passos, D. et al.: Minimum loss multiplicative routing metrics for wireless mesh networks. J. Internet Serv. Appl. **1**(3), 201–214 (2011)
23. Koksal, C.E., Balakrishnan, H.: Quality-aware routing metrics for time-varying wireless mesh networks. J. Sel. Areas Commun. **24**(11), 1984–1994 (2006)
24. Subramanian, A.P., Buddhikot, M.M., Miller, S.: Interference aware routing in multi-radio wireless mesh networks. In: 2nd Workshop on Wireless Mesh Networks, pp. 55–63 (2006)
25. Jiang, W. et al.: Optimizing routing metrics for large-scale multi-radio mesh networks. In: International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1550–1553 (2007)
26. Owczarek, P., Zwierzykowski, P.: Routing protocols in wireless mesh networks—a comparison and classification. In: Information Systems, Architecture and Technology, Network Architecture and Applications, pp. 85–95 (2013)
27. Campista, M., Elias, M.: Routing metrics and protocols for wireless mesh networks. Network **22**(1), 6–12 (2008)
28. https://omnetpp.org
29. https://github.com/inetmanet/inetmanet/wiki
30. Wasłowicz, M.: Routing in Wireless Mesh Networks. M.Sc. Thesis. Poznan University of Technology, Poznan (2014)

Energy Efficient Dynamic Load Balancing in Multipath TCP for Mobile Devices

Michał Morawski and Przemysław Ignaciuk

Abstract Over the last few years, one can observe a shift in preferences to use wireless media, including WiFi and cellular, rather than wired technologies for the communication purposes. On the other hand, a variant of TCP that allows for simultaneous transmission over different paths has been developed. This paper addresses the problem of optimal load distribution among the interfaces available at a networking device, e.g. mobile phone, with the objective of minimizing the overall energy consumption. A dynamic allocation algorithm, adapting to current, time-varying channel capacity is designed. Comparison with earlier approaches is provided and superiority of the proposed solution is demonstrated in numerical tests.

Keywords Multipath TCP · Load balancing · Energy efficiency

1 Introduction

Since its deployment three decades ago TCP continues to provide the core of communication capabilities in the Internet data transfer. In order to properly respond to new demands appearing over the years the protocol has evolved. The traditional TCP employs a single path (SPTCP) to effectuate data transfer even though multiple paths may be available in the network. Nowadays, servers have installed many network interfaces (NICs), use different routers, and client terminals are equipped with diverse connectivity solutions, e.g. Ethernet, WiFi, cellular, or WiMAX. However, due to the design restrictions of TCP, additional paths are employed only in case of failure. Multipath TCP (MPTCP) [1] has the potential to elevate the end-point transfer efficiency, and thus the user application performance,

M. Morawski (✉) · P. Ignaciuk
Institute of Information Technology, Lodz University of Technology, Lodz, Poland
e-mail: michal.morawski@p.lodz.pl

P. Ignaciuk
e-mail: przemyslaw.ignaciuk@p.lodz.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_15

by incorporating simultaneously multiple paths between the communicating parties. Even though the fundamental framework of MPTCP has been standardized [2], some research problems remain unresolved. One of such open research challenges is the development of methods of MPTCP stream splitting into multiple sub-streams, controlled by SPTCP.

The popularity of smartphones, tablets, mobile computers, and associated increase of their functional capabilities, have outgrown the progress in battery design. Those devices suffer from fast energy depletion rate which limits the overall user experience. The most energy-consuming part of hardware is the display, together with CPU/GPU efforts required to render the dynamic content of web-sites when staying on-line. Therefore, in order to improve the energy economy while effectuating the Internet connectivity, one should shorten the time of data transfer and allow the users to switch off the display. The paper proposes a new load-balancing solution, particularly well-suited for battery-powered devices equipped with a few independent interfaces. Although addressing similar problems as in the current literature [3–5], the idea presented here differs both in the objectives and in the way the traffic is distributed among the interfaces. It concentrates on specifying the operation of MPTCP scheduler. In the proposed approach, the way the MPTCP stream is separated into sub-streams by the scheduler is determined as a solution of the optimization problem formulated for the mathematical model of power consumption. The results of numerical comparison show that the presented—non-heuristic—method allows for substantial (as high as 50 %) savings in energy usage as compared to recent approaches.

2 Multipath Data Transfer

The considered architecture is illustrated in Fig. 1. The research objective is to design a load-balancing algorithm for the traffic scheduler. The scheduler acquires all the information necessary for the algorithm implementation from NICs and observation of the link layer activities.

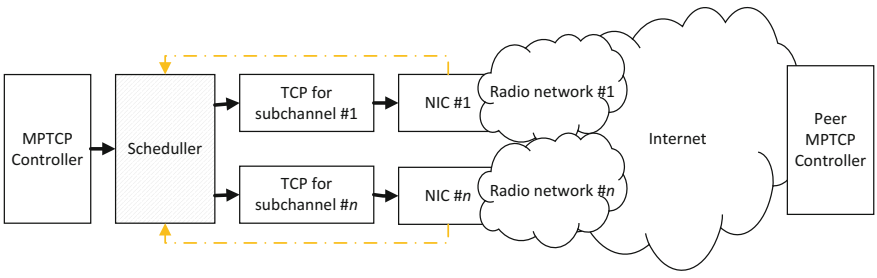


Fig. 1 MPTCP architecture

In the analyzed communication scenario, the user application seeks to send B bits of data to a remote peer. The data is placed in the TCP output buffer held by the local MPTCP controller. The MPTCP controller opens n sub-channels and distributes the traffic among the corresponding sub-streams that are controlled by the ordinary TCP. The ordinary SPTCP control is executed separately for each sub-channel. In the standardized MPTCP implementation, the unweighted round-robin algorithm is used, although other schedulers have also been tested [4]. In the subject literature, e.g. [6], one can find claims that the data distribution ratio in the load balancing procedure should be proportional to the sub-channels capacity. In this paper, it is shown that this claim is not always true.

The proposed solution incorporates the observation of NIC dynamic parameters (power consumption and throughput) to decide about the moment of extracting subsequent segments from the data buffer and to direct the segments to an appropriate sub-channel. The standard behavior of MPTCP (e.g. OILA) [7] and ordinary TCP (e.g. Cubic, Compound, Reno) [8] remains intact. The developed method governs the operation of the traffic scheduler.

3 Energy Considerations and Optimal Load Distribution

The display in mobile devices consumes more energy than the network interfaces. The NIC energy expenditure depends on transmission power, noise level, number of retransmissions, and control plane activity. These factors differ among interfaces and their mode of operation. In general, for WiFi the transmit power vary from 0 dBm to 10–15 dBm, and for the cellular interfaces from 10 dBm to 20–50 dBm [9, 10]. Additionally, certain amount of energy is dissipated by the NIC electronics [3].

In order to reduce the overall energy consumption, the following optimization problem can be considered: introduce an efficient load balancing method that minimizes E^o given by

$$E^o = \sum_{i=1}^{n} E_i + P_D \max_i T_i \tag{1}$$

subject to $T_i > 0$, where

$$E_i = \int_0^{T_i} p_i(t)dt > 0 \tag{2}$$

is the total energy dissipated by interface $i \in [1, n]$, $p_i(t) > 0$ is the power necessary to transmit data through this interface at time t, T_i is the total transmission time, and $P_D > 0$ is the power consumed by the display. $p_i(t)$ [W] is a function of the power efficiency of NIC—$\tilde{p}_i(t)$ (W/bit) and the number of bits sent through channel i—b_i:

$$p_i(t) = \tilde{p}_i(t) b_i + p_i^{op}, \tag{3}$$

where p_i^{op} is the power necessary to keep the NIC in the operational state. $\tilde{p}_i(t)$ and p_i^{op} are functions of:

- the distance from the end-point to the hub (access point, head-end, or BTS),
- the mode of operation, which includes selection standard "b", "g", "n", "ac", band, RTS, or CCA mode in WiFi and band, GPRS, or revision of UMTS class in the case of cellular interfaces.

$\tilde{p}_i(t)$ and p_i^{op} profiles can be obtained using the chipset datasheets as function of the parameters stated above, or can be directly measured by the end-point (such approach is suggested e.g. in [3]). The profile varies with time as a result of node mobility, or controller activity. Here, piece-wise constant evolution is assumed.

Since both the objective function and the constraints ($T_i > 0$) are convex, minimization of (1) is a convex problem [11]. In the nominal case, i.e. when $p_i(t) = p_i = const$, the problem may be solved analytically.

Lemma 1 *E^o is minimum for $T_1 = T_2 = \cdots = T_n$.*

Proof Assume $T_1 \geq T_2 \geq \cdots \geq T_n$. Then (1) can be expressed as

$$\begin{aligned} E^o =& p_1 T_1 + \sum_{i=2}^{n} p_i T_i + P_D T_1 = (p_1 + P_D) T_1 + \sum_{i=2}^{n} p_i T_i \\ =& (p_1 + P_D)(T_1 - T_2) + (p_1 + P_D) T_2 + \sum_{i=2}^{n} p_i T_i. \end{aligned} \tag{4}$$

By definition, all the elements in (4) are non-negative. Therefore,

$$E^o \geq (p_1 + P_D) T_2 + \sum_{i=2}^{n} p_i T_i, \tag{5}$$

i.e. when $T_1 - T_2$ vanishes from (4). Consequently, applying $T_1 = T_2$ in (5), one has

$$\begin{aligned} E^o &\geq (p_1 + p_2 + P_D)(T_1 - T_3) + (p_1 + p_2 + P_D) T_3 + \sum_{i=3}^{n} p_i T_i \\ &\geq (p_1 + p_2 + P_D) T_3 + \sum_{i=3}^{n} p_i T_i. \end{aligned} \tag{6}$$

Continuing this reasoning for subsequent T_i, one arrives at the minimum of E^o for $T_1 = T_2 = \cdots = T_n$. This conclusion ends the proof.

T_i can be modeled as $T_i = b_i / \bar{c}_i + \tau_i$ with $\bar{c}_i = \frac{1}{T_i} \int_0^{T_i} c_i(t) dt$ being the average capacity of channel i, $c_i(t)$—the capacity of channel i at time t, and τ_i—the

round-trip time at the end of transmission. Except for a very short transmission, $b_i/\bar{c}_i \gg \tau_i$, so $T_i \approx b_i/\bar{c}_i$. The granularity of b_i is neglected. On the other hand, $c_i(t) \le c_i^{\max}(t)$, where $c_i^{\max}(t)$ is the maximum capacity of channel i. With this notation, and setting $T_i = T$, the minimum of (1) can be expresses as

$$\begin{aligned} E^o_{\min} &= \sum_{i=1}^{n} p_i T + P_D T = \sum_{i=1}^{n} p_i T + \sum_{i=1}^{n} \frac{P_D}{n} T = \sum_{i=1}^{n} \left(p_i + \frac{P_D}{n} \right) \frac{b_i}{\bar{c}_i} \\ &= \sum_{i=1}^{n} \left(\tilde{p}_i b_i + p_i^{op} + \frac{P_D}{n} \right) \frac{b_i}{\bar{c}_i}. \end{aligned} \tag{7}$$

Let $B_j = B - \sum_{i=1}^{j-1} b_i$. Then, $b_n + b_{n-1} = B_{n-1}$, and using (3) and (7), the minimum energy consumed by these two channels can be evaluated as (assuming all the constraints are satisfied)

$$\begin{aligned} E^0_{n-1} &= \tilde{p}_n \frac{b_n^2}{\bar{c}_n} + \left(\frac{P_D}{n} + p_n^{op} \right) \frac{b_n}{\bar{c}_n} + \tilde{p}_{n-1} \frac{b_{n-1}^2}{\bar{c}_{n-1}} + \left(\frac{P_D}{n} + p_{n-1}^{op} \right) \frac{b_{n-1}}{\bar{c}_{n-1}} \\ &= \tilde{p}_n \frac{b_n^2}{\bar{c}_n} + \left(\frac{P_D}{n} + p_n^{op} \right) \frac{b_n}{\bar{c}_n} + \tilde{p}_{n-1} \frac{(B_{n-1} - b_n)^2}{\bar{c}_{n-1}} + \left(\frac{P_D}{n} + p_{n-1}^{op} \right) \frac{B_{n-1} - b_n}{\bar{c}_{n-1}}. \end{aligned} \tag{8}$$

The optimum is found by solving $\partial E^o_{n-1}/\partial b_n = 0$ for b_n, i.e.

$$b_n = \frac{\left(\frac{P_D}{n} + p_{n-1}^{op} \right) \bar{c}_n + \left(\frac{P_D}{n} + p_n^{op} \right) \bar{c}_{n-1}}{2\tilde{p}_n \bar{c}_{n-1} + 2\tilde{p}_{n-1} \bar{c}_n} - \frac{B_{n-1} \bar{c}_n \tilde{p}_{n-1}}{\tilde{p}_n \bar{c}_{n-1} + \tilde{p}_{n-1} \bar{c}_n}. \tag{9}$$

$b_{n-1} = B_{n-1} - b_n$ and subsequent b_i can be evaluated in the same way by substituting $b_{n-1} \to b_{n-2}$, $b_n \to B_{n-1}$ in (8).

The obtained solution is precise for the nominal operating conditions but impractical from the perspective of the actual communication process. Expression (9) is too complex for real-time implementation and the values need to be constrained to satisfy the assumption $0 \le b_i \le B_i$. Besides, some quantities (especially $\bar{c}_i$) are not accessible ahead of time and the optimal energy consumption and load balancing strategy would need to be learned a posteriori. For that reason, the online scheduler should be organized in a different—dynamic—way.

4 Dynamic Load Balancing

Solution (9) of problem (1) is obtained under the assumption of perfect knowledge of the communication system behavior. Moreover, if approximations are not applied, the optimal load distribution can be determined using numerical solvers

only, which is inadequate for real-time implementation. In order to circumvent these obstacles, a suboptimal solution will be introduced. In the paper, it is proposed to replace P_D by a term $\tilde{P}_D B$, where $\tilde{P}_D$ is the power efficiency of display. In the design, the power consumed by the display is assumed to depend on the volume of transmitted data. Additionally, p_i^{op} are neglected in the evaluation of power (3).

Introducing $\tilde{P}_i = \tilde{p}_i + \tilde{P}_D/n$, one may simplify (7) as

$$E^s = \sum_{i=1}^{n} \tilde{P}_i \frac{b_i^2}{\bar{c}_i}, \tag{10}$$

where E^s denotes the suboptimal energy consumption. Conducting the same reasoning as in (8) and (9), one obtains

$$\frac{b_n}{B} = \frac{\tilde{P}_{n-1}}{\bar{c}_{n-1}} \Big/ \left(\frac{\tilde{P}_{n-1}}{\bar{c}_{n-1}} + \frac{\tilde{P}_n}{\bar{c}_n} \right), \quad \frac{b_n}{b_{n-1}} = \frac{\tilde{P}_{n-1}}{\bar{c}_{n-1}} \Big/ \frac{\tilde{P}_n}{\bar{c}_n}, \tag{11}$$

and a good approximation of E^o when

$$\underset{i,j\in[1,n]}{\forall} \frac{b_i}{b_j} = \frac{\tilde{P}_j \bar{c}_i}{\tilde{P}_i \bar{c}_j}. \tag{12}$$

When $\tilde{P}_i = \tilde{P}_j$ (or $\tilde{P}_D \gg \tilde{p}_i$) and $\bar{c}_i = c_i^{\max}$, (12) coincides with the traditional way of traffic distribution. However, these assumptions are justified only when the end-point is equipped with homogenous and non-congesting interfaces (e.g. Ethernet). Generally, in the case of mobile devices, one should assume neither $\tilde{P}_i = \tilde{P}_j$ nor $\bar{c}_i = c_i^{\max}$.

5 How to Measure Sub-channel Capacity

The following method of measuring the current capacity of channel i is proposed. After a successful transmission of a data piece associated with the class of service selected by the user application at time t_a (including MAC procedure, scrambling, fragmentation, retransmissions, and acknowledgements), NIC signals that it is ready to send the next piece. The application inserts the data in the NIC queue. Just after *cwnd* value permits sending the next segment, the associated sub-stream prepares an m-bit size packet. The packet is placed in the NIC queue at time t_t. The corresponding frame is acknowledged at time t_{a+1}. The current channel capacity can be estimated as $c_i(t_{a+1}) = m/(t_{a+1} - t_t)$. The value of c_i fluctuates with time in the bounded range $(0, c_i^{\max}(t)]$. $c_i^{\max}(t)$ is approximately known and depends on

the link layer parameters of the associated interface, e.g. for CCA 802.11g, it varies from 40 % for long frames to about 10 % for short frames of the physical layer speed [9]. $c_i^{\max}(t)$ changes stepwise depending on SINR or RSSI and can be obtained from datasheets and norms [9].

6 Channel Capacity Prediction

The energy-efficient traffic distribution among NICs (12) requires the knowledge of average capacities, but these values are not known in real time. They can be determined by filtering the highly variable $c_i(t)$. The most common way to measure the channel capacity is through the low-pass filter $\hat{c}_i(t_{a+1}) = (1-\alpha)\hat{c}_i(t_a) + \alpha c_i(t_{a+1})$, typical in TCP implementations. Parameter $\alpha \in [0,1]$ influences the adaptation speed. Unfortunately, due to node mobility, it is possible to obtain $\hat{c}_i(t_{a+\Delta a}) > c_i^{\max}(t_{a+\Delta a})$ In such situation, the information about traffic intensity is erroneous, yet after some time, $\hat{c}_i(t)$ adjusts to new conditions. Unfortunately, the described case is not infrequent, as illustrated in Fig. 2, which presents typical RSSI fluctuations in a 3-min interval. The changes of power imply changes of $c_i^{\max}(t)$ [9] and frequent fluctuations of $c_i^{\max}(t)$ may cause the state of optimal distribution never to be reached.

Therefore, the following estimation of the average channel capacity is proposed:

$$\hat{c}_i(t_{a+1}) = \frac{c_i^{\max}(t_{a+1})}{c_i^{\max}(t_a)}[(1-\alpha)\hat{c}_i(t_a) + \alpha c_i(t_{a+1})]. \tag{13}$$

The scaling factor introduced in (13) allows for constraining the capacity estimate to a feasible interval.

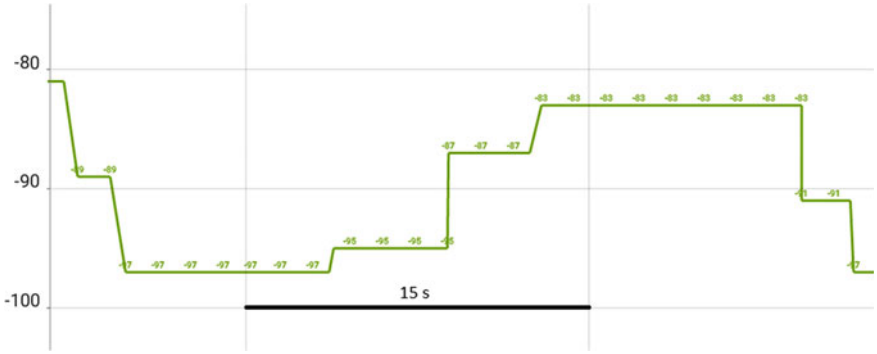


Fig. 2 Typical fluctuations of RSSI obtained from `GSMSignalMonitoring`

7 Experimental Evaluation

The algorithm is implemented using 'credits' as in common implementations of AQM. The on-line computational effort is negligible. In order to analyze the effectiveness of the proposed approach, several simulation experiments are conducted. In the tests, the MPTCP buffer holds 10,000 segments, each of the size of 1500 bytes. The data is to be sent through two channels with properties indicated in Fig. 3. The channels correspond approximately to WiFi (left) and UMTS (right) specifics. The display power $P_D = 20$ W. The purpose of experiments is to transmit content of the MPTCP buffer through both channels and compute the energy consumed by the end-point device. Table 1 presents the numerical values when only one channel is used (it corresponds to SPTCP) ((a)—only the first channel, (b)—only the second channel), implemented in the MPTCP official kernel round-robin scheduling (c), channel capacity proportional scheduling (d), and finally, proposed power-efficient scheduling when P_D is neglected (e), and its full version (P_D included) (f). The epoch is the time between taking subsequent scheduler decisions.

The data gathered in Table 1 shows that MPTCP is in general not an energy-efficient solution for mobile devices (c). It does not outperform SPTCP using a 'greener' technology, e.g. WiFi (a) over UMTS (b). In the considered case, SPTCP results in approximately 3 times lower battery depletion than the standard round-robin MPTCP implementation—(a) vs. (c). In turn, the proposed algorithm (e), and particularly in its full version (f), allows for more than 50 % energy savings at a negligible computational cost giving 20 % lower energy consumption than the capacity proportional scheduling (d).

The results displayed in Table 1 are obtained for the nominal channel capacities, i.e. the capacity measurement is influenced neither by fluctuations, nor by delays and (13) responds with the maximum value. Therefore, another series of experiments are conducted for a more realistic situation where the capacities vary according to the Poisson process with $\lambda = 4$, scaled to $0.33c_i^{\max}(t)$ and longer on-off traffic variations (also scaled to $0.33c_i^{\max}(t)$) complemented by slowly-changing trends, depicted in Fig. 4.

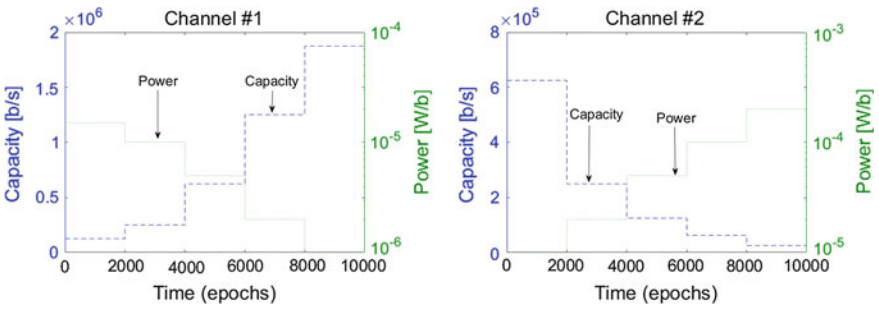


Fig. 3 Channel capacity and power profile (in logarithmic scale)

Table 1 Comparison of different MPTCP scheduling methods

	a	b	c	d	e	f
T_1 (s)	43.01	0	22.4	17.87	19.52	16.04
T_2 (s)	0	208.8	104.4	17.92	8.83	11.28
E_1 (J)	95.04	0	49.5	43.67	48.98	41.32
E_2 (J)	0	1140	570	102.50	50.10	63.29
b_1 (kB)	15,000	0	7500	10,315.5	11287.5	10603.5
b_2 (kB)	0	15,000	7500	4684.5	3712.5	4396.5
Total energy (J)	955.2	5316	2708	504.66	489.58	425.37

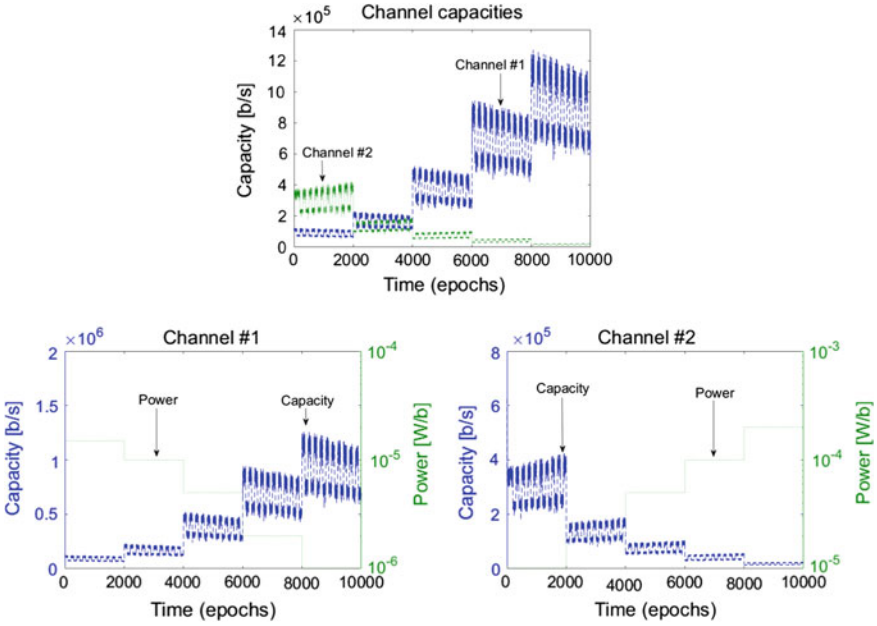


Fig. 4 Channel capacity and power profile for congested channels: real profile—upper graph, estimated profiles—lower graphs

The channel capacity is averaged by low-pass filter (13) using different α values and different delays. Table 2 presents the relative impact of these parameters on the overall device energy consumption. As a reference, $\alpha = 1$ (no filtering) and delay = 1 epoch (marked) is applied. Because the proposed algorithm uses "the best" channel to full extent, and only supplements the communication by "worse" ones, the delay between subsequent evaluation of the "worse" channels increases (assuming no other traffic is present). For such channels the correct results require averaging using different smoothing coefficients. This property can be employed in the design of a control strategy for the simultaneous download process—a next step in the research work on the subject.

Table 2 Influence of averaging coefficient on algorithm effectiveness

Delay/α	1	10	50	100	500
1	1	1.01	1.04	1.08	1.07
1/2	1.01	1.01	1.03	1.05	1.03
¼	1.01	1.02	1.03	1.03	0.99
1/8	1.02	1.03	1.03	1.03	0.96
1/16	1.02	1.02	1.02	1.01	0.94

8 Summary and Conclusions

The paper presents a new algorithm for MPTCP traffic scheduler. It distributes the load over the network interfaces of mobile, battery-powered devices, like tablets or smartphones, to achieve low energy consumption. The algorithm is formulated on the basis of a model-based optimization problem and mathematical analysis of channel properties. It dynamically adjusts the channel usage depending on their current characteristics (congestion at the first hop and distance from the hub). The conducted simulation experiments show that the proposed solution allows for as high as 50 % decrease in the consumed energy as compared to the typically applied round-robin scheduling. Further work on the problem will be focused on improving the energy efficiency while concurrent upload and download operations are in progress.

Acknowledgments P. Ignaciuk holds a scholarship of the Polish Ministry of Science and Higher Education for outstanding young researchers.

References

1. Peng, Q., Walid, A., Hwang, J., Low, S.H.: Multipath TCP: analysis, design, and implementation. IEEE/ACM Trans. Netw. **24**(1), 596–609 (2016)
2. Ford, A., Raiciu, C., Handley, M., Bonaventure, O.: TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, IETF (2013)
3. Deng, S., Netravali, R., Sivaraman, A., Balakrishnan, H.: WiFi, LTE, or both? Measuring multi-homed wireless internet performance. In: Proceedings ACM IMC, Vancouver, pp 181–194 (2014)
4. Paasch, C., Ferlin, S., Alay, O., Bonaventure, O.: Experimental evaluation of multipath TCP schedulers. In: Proceedings of ACM SIGCOMM CSWS, Chicago, pp 27–32 (2014)
5. Hwang, J., Yoo, J.: Packet scheduling for multipath TCP. In: Proceedings of 7th International Conference on Ubiquitous and Future Networks, Sapporo, Japan, pp. 177–179 (2015)
6. Park, S.Y., Joo, C., Park, Y., Bank, S.: Impact of traffic splitting on the delay performance of MPTCP. In: Proceedings of IEEE International Conference on Communications (ICC), Sydney, Australia, pp. 1204–1209 (2014)
7. Khalili, R., Gast, N., Popovic, M., Boudec, J.-Y.L.: MPTCP is not pareto-optimal: performance issues and a possible solution. IEEE/ACM Trans. Netw. **21**(5), 1651–1665 (2013)

8. Abdeljaouad, I., Rachidi, H., Fernandes, S., Karmouch, A.: Performance analysis of modern TCP variants: a comparison of Cubic, Compound and New Reno. In: Proceedings of 25th biennial symposium on communications (QBSC), Kingston, pp. 80–83 (2010)
9. Gast, M.: 802.11ac: A Survival Guide. O'Reilly, Sebastopol (2013)
10. Hämäläinen, S., Sanneck, H., Sartori, C. (Eds.) LTE Self-organizing Networks (SON). Network Management Automation for Operational Efficiency. Willey, Chichester (2012)
11. Boyd, S., Vandenberghe, L.: Convex Optimization, 7th edn. Cambridge University Press, Cambridge (2009)

Part V
Data Processing Tools

Mutation Testing in Model Accuracy Assessment

Joanna Strug

Abstract Abstract models built during a development of software systems play an important role in producing a high quality system. Any modeling mistakes, if not corrected, will propagate to the further development stages decreasing the quality of the final system and increasing costs of correcting them. It is of primary importance to make sure that the model conforms to all requirements of the stakeholders and ensures proper work of the future system under various conditions. This paper describes a mutation testing based approach to accuracy assessment of conceptual models built at the beginning of a system development. The approach focuses on providing test cases for assessing and measuring accuracy of such model with respect to its ability to handle unexpected and erroneous situations. Mutation testing is usually used to assess quality of test cases, but it can also help to provide, in a systematic and human-unbiased way, a number of test cases representing wide range of unexpected situations.

Keywords Model evaluation · Mutation testing · Software modeling · UML · OCL

1 Introduction

Abstract models, focusing on different characteristics of a system to be designed, are built at all stages of the system development process [1]. Such model, if being accurate reflections of system characteristics of interest for a particular level of abstraction, contributes to successful development of a flawlessly working system. However, modeling involves taking decisions regarding elaboration and even refinement of requirements, definition of the system structure and operations, selection of components performing these operations, and so on. As modeling

J. Strug (✉)
Faculty of Electrical and Computer Engineering, Cracow University of Technology, Kraków, Poland
e-mail: pestrug@cyf-kra.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_16

activities are errors prone, accuracy of models cannot be taken for granted. Thus, time and effort should be invested into developing models conforming to requirements of the stakeholders and ensuring proper work of the system under various conditions [2].

Development of a model can be seen as an iterative process involving, after building an initial model, iterative assessment of the model accuracy and its improvements. The assessment step is essential in determining the degree to which the model fulfills the expectations and in providing useful data for its improvement.

Main causes of inaccuracy of a conceptual model include missing and incorrectly modeled requirements, as well as lack of proper handling of unexpected, erroneous situations and data. Faults caused by any of the first two of them can usually be detected by positive test cases reflecting, in a representative way, the intended ways of using the system that are given by a system specification [3]. However, due to the fact that early prediction of unexpected and potentially erroneous situations is limited to some obvious cases, the test cases ability to check if a model can handle them properly is very limited [4]. Thus, an effective approach to an assessment of a model accuracy should also provide a subset of negative test cases [3] designed specifically for checking how the model handles the unexpected situations.

A general approach to mutation testing based negative testing was presented in [5] and in [6] a concept of applying mutation testing to assess a model accuracy, focusing on this second aspect of the accuracy, was introduced. This paper builds on the concept by adding more details, shows how to use information coming from the evaluation to improve a model, and presents a real life case study.

The paper is organized as follows. Section 2 briefly presents background information and related works. The approach to an evaluation of a model accuracy is described in details in Sect. 3, and Sect. 4 presents the case study. In the last section conclusions and directions for future works are given.

2 Background and Related Work

A great number of research deals with various problems related to software testing, but only some of them regard negative testing [3, 7–9]. Techniques, such as equivalence partitioning [8] or stress testing [7, 9] provide advises on selecting test cases with invalid input values or on creating and testing some extreme conditions for the system. However, test cases selected by mean of such techniques may not be able to reflect a wide range of unexpected, but not necessarily extreme, situations.

Mutation testing, a technique that in general serves the derivation of faulty artifacts from correct ones, seems to be adequate for contributing to the solving the problem of providing negative test cases. The technique was originally introduced to evaluate quality of test suites provided for programs [10]. Its application involved generation of a number of faulty version of a correct program (called mutants) by introducing small syntactic changes into the code of the original program and then

running the mutants with tests from the evaluated test suite. The ratio of the number of mutants detected by the tests over the total number of non-equivalent mutants generated for the original program (called a mutation score), determined the quality of the evaluated test suite in terms of its ability to detect faults.

A test quality assessment is still the main application area of mutation testing, but works showing its application at different levels of abstractions and to different artifacts are more and more common (examples of such works are given in [11]). However, papers on using mutation testing for deriving negative test cases are still rare. To the author's best knowledge only a few other researchers have also studied this topic [12–14]. The authors of the work in [13] mutated a model of a system and used model-checking techniques to generate counter-examples showing violation of certain properties. The counter-examples made a set of negative test cases, but due to the fact that mutants of the model represent incorrect behaviour of the system, some of the test cases may in fact detect specification related inconsistencies that can also be detected by positive test cases. Another example of an approach, where tests cases were derived from mutated model was presented in [14]. The approach described in [12] is the most relevant to the one presented in this paper, as it also aims at modifying tests directly. However, the modifications proposed by the authors are random and affect only data processed by a program. The approach presented in [5] and the one presented here provide much wider range of changes introduced in a controlled way by using a set of mutation operators defined by the author specifically to target test cases [5, 6].

3 An Approach to Model Accuracy Assessment

A conceptual model is built upon requirements that specify what a system is expected to do, what kind of data it should process, under which conditions it should operate, and so on. The requirements do not provide a clear answer to a question of how the system should behave when subjected to conditions and data out of its normal scope of operations. Issues arising from such a question need to be resolve during development of the conceptual model, otherwise they will propagate to further development stages and finally would make the final system vulnerable to failures.

Development of an accurate model, handling properly various unexpected and erroneous situations, is not easy, because a developer's ability to predict situations that may lead to failures is rather limited. Thus, it is necessary to run the model with negative test cases being able to trigger a wide range of unexpected use scenarios for the system and observe how the model behaves in these situations. Unfortunately, test cases selected basing on requirements are positive, that is they represent only intended ways of using the system and do not support detection of this kind of problems.

The approach presented here focuses on assessing a model, during its development, with the aim to detect its weaknesses caused by lack of proper handling of

unexpected situations. It deals with this problem by providing and using negative test cases that trigger such situations. The main idea behind this approach is to use mutation testing to generate a number of test cases, each being a modified version of any of the positive test cases. Each mutant is a negative test case, as it represents some unintended use scenario for the system. The model, when run with a mutant, will either behave as if the mutant represented some expected use scenario, or it will fail showing its ability to recognize the fact that the mutant triggered an unexpected way of using the system.

The following subsections outline the approach to a model evaluation and describe its main stages in details.

3.1 An Outline of the Approach

Before the assessment of a conceptual model starts the model, representing requirements defined for a system, and a suite of test cases selected basing on these requirements, have to be provided. The assessment is carried out in three stages (Fig. 1):

1. generation of mutated test cases,
2. execution of mutated test cases, and
3. analysis of assessment results.

The first stage involves generation of mutated test cases. The mutants are obtained by introducing small changes into the original test cases accordingly to a predefined rules, called mutation operators. Then, in the second stage, each mutant is executed against the model and a verdict (accepted or rejected) is assigned to it. Finally, in the last stage, accuracy of the model is calculated basing on the number of mutants accepted and rejected by the model and the mutants are analyzed to provide feedback for improving the model.

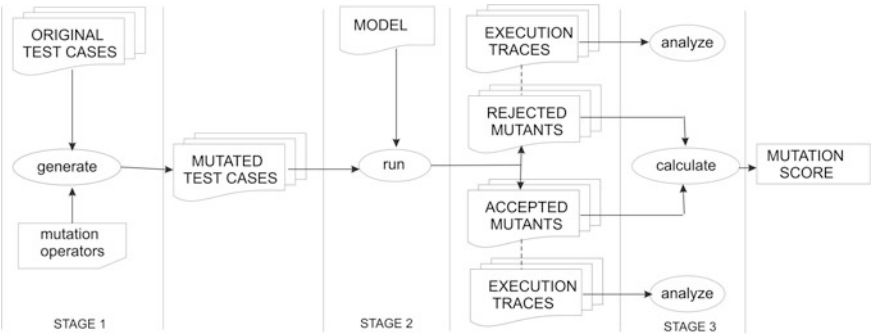


Fig. 1 Outline of the approach

3.2 Stage 1—Generation of Mutated Test Cases

The expected input artifact for the first stage is a suite of positive test cases representing all intended ways of using a system, as given by the requirements. A single test case should specify conditions, data and steps (usually represented by operations calls) needed to force a model to behave in a particular way, as well as expected results that the model should produce after executing the test case.

To generate mutants of the test cases a set of mutation operators is needed. Here, a mutation operators is a rule specifying an element of a test case that can be changed and defining the ways it can be changed. A set of mutation operators applicable to test cases consists of the following 9 operators [5, 6]:

- Condition Part Deletion (CPD)—deletes a part of a condition expected to hold before executing a test case,
- Condition Part Replacement (CPR)—replaces a part of a condition by another, compatible element of a condition,
- Operation Call Deletion (OCD)—deletes an operation call (i.e. a step) from a sequence of operation calls in a test case,
- Operation Call Replacement (OCR)—replaces an operation call in a sequence of operation calls in a test case by another operation call,
- Operation Call Insertion (OCI)—inserts an extra operation call in a sequence of operation calls,
- Operation Call Swap (OCS)—changes the order in which two subsequent operation calls should be performed,
- Operation Parameter Replacement (OPR)—replaces a value of a parameter in an operation call (i.e. data passed within a given step) with another, compatible value,
- Operation Parameter Swap (OCS)—changes the order of two values of parameter in a given operation call,
- Operation Target Replacement (OTR)—replaces a target of an operation call with another one.

The mutation operators cover all elements of a typical test case, so the set can be seen as sufficient for generating mutants triggering wide range of unintended ways of using a system. Algorithms outlined in [5] can be used to generate the mutants.

3.3 Stage 2—Execution of Mutated Test Cases

The expected input artifacts for the second stage are: the set of mutants generated in the first stage and the model of system undergoing the assessment.

In general the model may represent a system at any level of abstraction, but in this approach a focus is on a conceptual model built at an early stage of the system development. It is therefore expected that the model represents requirements

specified for the system by giving conceptual architecture and functionality of the designed system. The model has to be executable, thus the formalism used to describe the model should include means to define processing. The model, before being assessed with respect to its handling of unexpected situations, should be tested with all original test cases to ensure that the specified requirements have been modeled correctly.

Once the prerequisites concerning the model are satisfied the model is run with each mutated test case. When a mutant is executed its execution trace is generated and linked to the mutant, and after it has been executed a verdict is assigned to the mutant. A verdict rejected is assigned to a mutant when the model fails, otherwise a verdict accepted is assigned to the mutant.

3.4 Stage 3—Analysis of Assessment Results

The third stage uses the results obtained in the second stage: the verdicts assigned to mutants and the execution traces of mutants, to determine the accuracy degree of the assessed model and to provide feedback for refining the model.

The accuracy of a model, in term of its ability to recognized unexpected situation [6], is indicated by a mutation score. Let's for the rest of this paper M denotes the assessed model, T' denotes the set of mutants generated for a set of positive test cases denoted by T, T'_A and T'_R denote subsets of T' consisting of accepted and rejected mutants respectively, and T'_E denotes a subset of equivalent mutants [11].

The mutation score (denoted by MS) for a model M is defined as follows [Eq. (1)]:

$$MS(M) = \frac{|T'_R|}{(|T'_A| + |T'_R|) - |T'_E|} \tag{1}$$

where:

$|T'_A|$ is the number of mutants accepted by the model M,
$|T'_R|$ is the number of mutants rejected by M, and
$|T'_E|$ is the number of equivalent mutants.

The mutation score expresses, in a quantitative way, the degree of a model accuracy. Basing on the value of mutation score one can decide whether the development of the model should be continued or may be finished. The highest possible value of mutation score is 1—it means that all mutants were rejected.

When the mutation score implies that the model accuracy is not acceptable yet, the mutants and their execution traces should be analyzed. An execution trace of a mutant shows, in details, what the model really did when it was run with the mutant. So, an examination of an execution trace of a mutant should help to find out why the mutant was rejected or accepted and prepare a report describing the results of examination.

Rejection of a mutant shows that an evaluated model poses the ability to recognize an unexpected situation defined by the mutant and fails to work. Thus, an examination of an execution trace of the rejected mutant helps to identify the operations that drove the model to fail and to decide (possibly together with stakeholders) what the proper handling of a class of erroneous situations represented by the mutant should be like.

When an evaluated model accepts a mutant, it shows that the model is not able to recognize an unexpected situation and works further producing some, seemingly correct, results. Examination of an execution trace of the accepted mutant helps to identify faulty constraints specifying applicability range of operations performed when the model is run with the mutant, and thus suggests ways of improving the model.

Once, a model has been improved accordingly to the suggestions, it should be assessed again to see if the mutation score has reached an assumed level.

4 Case Study: A HVAC System

The case study demonstrates application of the described approach to evaluation of conceptual models on a software controlling a Heating, Ventilation and Air Conditioning system (HVAC) [15].

In general, the HVAC system maintains the room temperature within an assumed range. A user of the system can turn it on and off and set the temperature range. The system, when turned on, displays its current status, check the temperature in the room and, basing on the current value of the temperature, cools or heats the room.

The approach, as described in Sect. 3 can be applied to assess models described by means of various formalisms, but this work is aimed at models of object-oriented systems. Thus, UML/OCL class diagram was used to model the system [16, 17]. The class diagram describes structure of a system by giving elements (classes) of the system, their properties (attributes), functionalities (operations) and relations between these elements [16], as well as constraints specifying the operations in the form of their pre-conditions and post-conditions [17]. The pre- and post-conditions, are here of particular importance. They define conditions that should hold before an operation starts and when it ends, respectively. Missing or incorrectly specified pre- and post-conditions make the model vulnerable to unexpected behavior. The class diagram representing structure of the HVAC system is given in Fig. 2. The constraints defined for the system are not depicted in this figure for clarity, but an example pre-condition of the operation *regulate()* in class Controller is shown as an annotation in the Fig. 2.

A suite of test cases, the second element required by the assessment approach, was prepared manually. In the case study the suite consisted of only one test case, shown in Fig. 3a. It is given in a format required by the USE simulator [18].

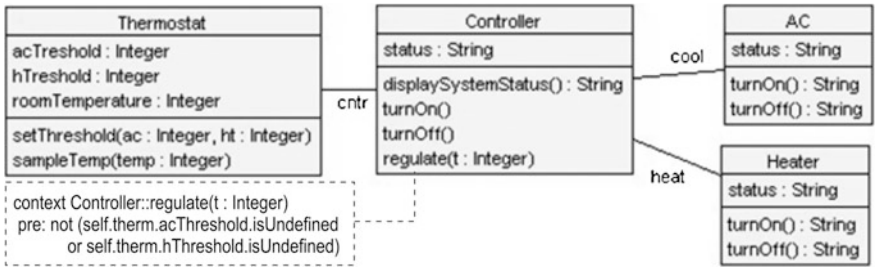


Fig. 2 A class diagram for HVAC system (screenshot from USE)

(a)

```
!therm:=new Thermostat
!cntr:=new Controller
!ac:=new AC
!ht:=new Heater
!insert (therm,cntr) into cntr
!insert (cntr, ac) into cool
!insert (cntr, ht) into heat
!cntr.status:='off'
!ac.status:='off'
!ht.status:='off'

!cntr.turnOn()
!therm.setThreshold(28,18)
!therm.sampleTemp(15)
!therm.sampleTemp(25)
!therm.sampleTemp(30)
!therm.sampleTemp(25)
!cntr.turnOff()
```

(b)

```
!therm:=new Thermostat
!cntr:=new Controller
!ac:=new AC
!ht:=new Heater
!insert (therm,cntr) into cntr
!insert (cntr, ac) into cool
!insert (cntr, ht) into heat
!cntr.status:='off'
!ac.status:='off'
!ht.status:='off'

!cntr.turnOn()
!therm.setThreshold(28,18)
!therm.sampleTemp(15)
!therm.sampleTemp(30)
!therm.sampleTemp(30)
!therm.sampleTemp(25)
!cntr.turnOff()
```

OPR

(c)

```
!therm:=new Thermostat
!cntr:=new Controller
!ac:=new AC
!ht:=new Heater
!insert (therm,cntr) into cntr
!insert (cntr, ac) into cool
!insert (cntr, ht) into heat
!cntr.status:='off'
!ac.status:='off'
!ht.status:='off'

!cntr.turnOn()
```

OCD

```
!therm.sampleTemp(15)
!therm.sampleTemp(25)
!therm.sampleTemp(30)
!therm.sampleTemp(25)
!cntr.turnOff()
```

Fig. 3 **a** An example test case t for HVAC, **b** and **c** examples of mutants for t

4.1 Generation of Mutants

The test case (denoted by t) used in the case study describes a scenario for checking if the model makes it possible to turn on and off the system, and to turn on and off the Heater and the AC, when the temperature rises and then falls down.

For the test case 51 mutants were generated. Detailed information on the number of mutants generated by applying each of the mutation operators is given in the second column of Table 1. Two of the mutants generated for t are shown in Fig. 3b, c.

4.2 Execution of Mutants

The mutants generated for test case t were executed within a USE environment [18]. The USE allows to run a model with a specified test case by simulating calls of the model operations (according to a scenario provided by the test case). While running the model, the tool checked if the pre- and post-conditions of called operations were

Table 1 Statistics by operators for mutants generated for t

Mutation operator	# of mutants for t ($\|T'\|$))	# of rejected mutants ($\|T'_R\|$)	# of accepted mutants ($\|T'_A\|$)	# of equivalent mutants ($\|T'_E\|$)
CPD	3	0	3	0
CPR	5	3	2	0
OCD	6	1	5	2
OCR	7	2	5	4
OCI	14	5	9	2
OCS	5	2	3	1
OPR	10	0	10	5
OPS	1	0	1	0
OTR	0	0	0	0

satisfied. An unexpected situation represented by a mutated test case was recognized if a violation of any condition was reported or the model failed. After executing a mutant, depending on the simulation results, a verdict rejected or accepted was assigned to it. Table 1 summarizes results of running the mutants. It gives, for each mutation operator, the number of accepted, rejected and equivalent mutants.

The results of executing mutants were presented within the USE in a form of a sequence diagrams [16]—these diagrams show the execution traces for the mutants.

4.3 Analysis of the Assessment Results

While most activities of the two previous stages of the model assessment can be automated, the analysis has to be performed manually.

First the accepted mutants were analyzed to identify and remove the equivalent mutants and the mutation score for the initial model was calculated as given by Eq. (1). Here, a mutation score of 0.35 was reached, what indicates rather low accuracy of the initial model and need for improvements.

To provide a feedback for the necessary improvements the execution traces of the remaining accepted mutants were analyzed to find out why they were accepted and what the model did wrong. For example, the mutant given in Fig. 3b was accepted. It represents a situation when the room temperature rises, in an instance, from below the lower threshold to above the upper threshold. While such situation seems unlikely to occur it is not entirely impossible. The examination of the mutant showed that the model turned on the AC, but it did not turn off the Heater. It is clearly an erroneous behavior of the model that needs to be fix, as both the Heater and the AC should never be working at the same time.

The rejected mutants were also analyzed to identify the operation that caused the model to fail. The mutant from Fig. 3c was rejected. It represents a situation when the user did not set the thresholds. The precondition defined for the regulate()

operation was violated, so the model was able to recognized erroneous situation but the model developer (or stakeholders) should still propose some solution that would allow the model to handle this situation.

Although the analysis performed in this stage are quite laborious, especially in case of accepted mutants, a careful inspection of the execution traces of mutants helps to gather information that in turn helps to improve the model, and thus the final systems that is to be developed basing on the model.

5 Conclusions and Future Work

A high quality system should always work flawlessly, thus it should never, in any situation, provide incorrect results and should be able to manage some unexpected situation without crashing, so early modeling of proper handling of such situations will significantly contribute to improvement of the quality of the final system. The approach presented in this paper contributes to the domain of developing models, as it provides a way to assess their accuracy with regard to their ability to recognized erroneous situations. This aspect of a model accuracy is rarely considered [13].

The approach uses mutation testing to generate, in a systematic and human-unbiased way, a number of test cases being able to trigger a wide range of unexpected situations. While mutation testing is a very effective assessment technique, it is also quite expensive in term of costs of generating and executing mutants [11]. Several cost reduction techniques were proposed, so far (a survey is given in [19]). A study on applying such techniques, especially the selective and structure dependant ones [20–24], in this context should also be a part of future work. Another problem that needs to be addressed is the identification of equivalent mutants [25]. It may also be worth to explore the possibility to use higher order mutations [11, 25] that could help to overcome both above problems.

In this paper the approach was studied in the context of models represented by means of UML and OCL. Still, the general approach to mutation testing based negative testing [5] may be easily adapted for other modeling formalisms or to other levels of system descriptions, as the general structure of a test cases will remain unchanged.

References

1. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers, San Rafael (2012)
2. Schamai, W., Helle, P., Fritzson, P., Christiaan, J., Paredis, J.: Virtual verification of system designs against system requirements. In: Models in Software Engineering. LNCS, vol. 6627, pp. 75–89. Springer, Heidelberg (2010)
3. Roman, A.: Testing and Software Quality. PWN, Warsaw (2015) (in polish)

4. Fernandez, J.-C., Mounier, L., Pachon, C.: A model-based approach for robustness testing. In: Testing of Communication Systems. LNCS, vol. 3502, pp. 333–348. Springer, Heidelberg (2005)
5. Strug, J.: Mutation testing approach to negative testing. J. Eng. **2016**, 13 p. (2016)
6. Strug, J.: Mutation testing approach to evaluation of design models. Key Eng. Mater. **572**, 543–546 (2014)
7. Briand, L.C., Labiche, Y., Shousha, M.: Stress testing real-time systems with genetic algorithms. In: Conference on Genetic and Evolutionary Computation, pp. 1021–1028, Washington, DC (2005)
8. Reid, S.C.: An empirical analysis of equivalence partitioning, boundary value analysis and random testing. In: International Software Metrics Symposium, pp. 64–73, Albuquerque, NM (1997)
9. Zhang, J., Cheung, S.C.: Automated test case generation for the stress testing of multimedia systems. Softw. Pract. Exp. **32**, 1411–1435 (2002)
10. DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on test data selection: help for the practicing programmer. Computer **11**, 34–41 (1878)
11. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE Trans. Softw. Eng. **37**, 649–678 (2011)
12. Bolazar, K., Fawcett, J.W.: Measuring component specification-implementation concordance with semantic mutation testing. In: International Conference on Computers and Their Applications, pp. 102–107, New Orleans (2011)
13. Fraser, G., Wotawa, F.: Using model-checkers for mutation-based test-case generation, coverage analysis and specification analysis. In: International Conference on Software Engineering Advances, pp. 16–21, Tahiti (2006)
14. Belli, F., Budnik, C.J., Hollmann, A., Tuglular, T., Wong, W.E.: Model-based mutation testing—approach and case studies. Sci. Comput. Program. **120**, 22–48 (2016)
15. Bahill, T., Daniels, J.: Using objected-oriented and UML tools for hardware design: a case study. Syst. Eng. **6**, 28–48 (2003)
16. Unified Modeling Language http://www.omg.org/spec/UML/2.5
17. Object Constraint Language http://www.omg.org/spec/OCL/2.4
18. Gogolla, M., Buttner, F., Richters, M.: USE: a UML-based specification environment for validating UML and OCL. Sci. Comput. Program. **69**, 27–34 (2007)
19. Usaola, M.P., Mateo, P.R.: Mutation testing cost reduction techniques: a survey. IEEE Softw. **27**, 80–86 (2010)
20. Strug, J.: Classification of mutation operators applied to design models. Key Eng. Mater. **572**, 539–542 (2014)
21. Ammann, P., Delamaro, M.E., Offutt, J.: Establishing theoretical minimal sets of mutants. In: IEEE International Conference on Software Testing, Verification and Validation, pp. 21–30, Cleveland Ohio, USA (2014)
22. Strug, J., Strug, B.: Machine learning approach in mutation testing. In: Software and Systems. LNCS, vol. 7641, pp. 200–214. Springer, Heidelberg (2012)
23. Strug, J., Strug, B.: Using structural similarity to classify tests in mutation testing. Appl. Mech. Mater. **378**, 546–551 (2013)
24. Strug, J., Strug, B.: Classifying mutants with decomposition kernel. In: Artificial Intelligence and Soft Computing. LNCS, vol. 9692, pp. 644–654. Springer, Heidelberg (2016)
25. Madeyski, L., Orzeszyna, W., Torkar, R., Józala, M.: Overcoming the equivalent mutant problem: a systematic literature review and a comparative experiment of second order mutation. IEEE Trans. Softw. Eng. **40**, 23–42 (2014)

Generating Source Code Templates on the Basis of Unit Tests

Mariusz Nyznar and Dariusz Pałka

Abstract This paper describes the solution for supporting Test-driven development (TDD) methodology with the help of code analysis and generation. By automation of the creation of stub implementation, it helps to accelerate the process of passing a red phase of Red/Green/Refactor (RDR) cycle. The tool described in the paper performs a semantic analysis of the test methods defined in a subjected Java class. It uses the information gathered during a process for further generation of unit stubs. The proposed solution describes the way in which the Abstract Syntax Tree (AST) is analyzed by the statements tree decomposition. The paper also presents possible ways of preventing and resolving unit types conflicts. The tool used for stub generation of units was defined in one of the test methods implemented in a *Log4J* Java framework. The results obtained are compared to a real class definition, and possible ways for further extensions of the solution are suggested.

Keywords Code generation · Test-driven development · Abstract syntax tree · Semantic analysis

1 Introduction

Test-driven development (TDD) is a software development methodology in which creating a fragment of a code is preceded by creating a test for this particular fragment. In its current form, TDD originated in Extreme Programming (XP) software development paradigm, and, since 2003, when it was rediscovered by Ken Back [2], it has been widely used in agile software development methods.

M. Nyznar (✉) · D. Pałka
AGH University of Science and Technology, 30 Mickiewicza Av, 30-059 Kraków, Poland
e-mail: mnyznar@student.agh.edu.pl

D. Pałka
e-mail: dpalka@agh.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_17

A software development process in TDD is based on the red/green/refactor cycle (RGR cycle) [2], which consists of the following steps [7] (see Fig. 1):

1. **Think**—On the basis of the predictions regarding the behaviour of the code, a test is planned which fails until this behaviour is obtained.
2. **Red bar**—A short test is written which requires the code to perform the desired behaviour. The test is written under the assumption that the code already exists. Because the desired behaviour is not present in the code yet, running this test will show errors indicated by the red progress bar in typical testing tools.
3. **Green bar**—Enough production code is written to make the test pass. After the test is run in testing tools, the success will be typically indicated by the green progress bar.
4. **Refactor**—During this step improvements are introduced into the design and the implementation of the program—because now the tests pass, the refactor of the code can be made without worrying about spoiling anything.
5. **Repeat**—The cycle is repeated to add new behaviour to the program.

For statically typed compiled languages, for example Java, in step 2 ('Red bar') two additional phases can be distinguished:

- **Non-compiling phase**—the program contains compilation errors because the method for obtaining the desired behaviour is not present in the production code but is called from the test code.
- **Compiling phase with errors in tests**—after adding the production code stub, the code can be compiled but tests indicate errors because the desired behaviour has not been implemented yet (only the code stub is present).

This paper demonstrates the implemented tool which makes it possible to automatically (or semi-automatically) generate the code stubs on the basis of prepared unit tests for Java language. So, this tool can significantly accelerate working

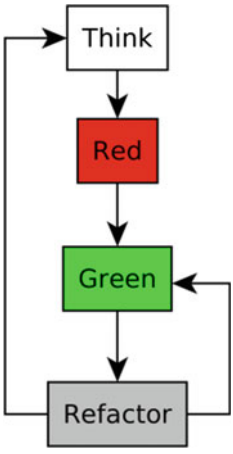


Fig. 1 RGR Cycle

of the RGR cycle by automatically passing from the 'non-compiling' phase to the 'compiling phase with errors in tests'.

2 Gathering Information from Unit Tests

All data required to generate a source code templates are obtained during the code analysis process. As a base for the analysis there are used a formal grammars. Code analysis can be explained as a transformation from the text form of the source code, to some hierarchical structure, e.g. Abstract Syntax Tree (AST) [5]. This process can be divided into the 3 main parts—a lexical, syntactic and semantic analysis. The first one is used to convert input defined as a code, to a lexical symbols. These are used as a base for the syntactic analysis, which produces the AST. The last part uses the AST to acquire information required for further code generation.

2.1 Syntactic Analysis

The syntactic analysis process uses the sequence of tokens to build a syntax tree, conforming to the set of production rules, defined by analyzed formal grammar of a language.

There are many parsing methods, like a Cocke-Younger-Kasami [6], or Earley's [3] algorithm, which can be used for any context-free grammar [1]. However, these methods are not commonly used because of their low efficiency. The most commonly used types of syntactic analysis are Top-down and Bottom-up parsers. Their names indicates the direction of the analysis. The most efficient methods for these types use LL and LR grammar subclasses [1].

The structure of the AST depends on the type of the analyzed instruction. For example, the structure of the variable declaration in Java language is presented in Fig. 2.

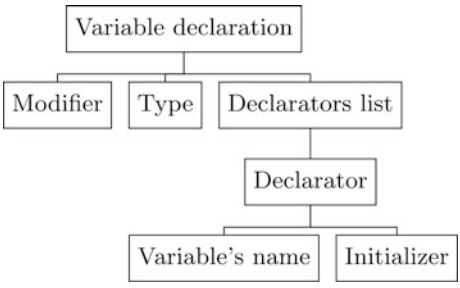


Fig. 2 The structure of variable declaration. Figure on the basis of [4]

2.2 Semantic Analysis and Generating Source Code Templates

The process of the semantic analysis is used to check semantic correctness of the program's code instructions. A semantic verification based on the AST structure is created as a result of parsing. The typical parts of static control are [1]:

- Types check,
- Program's flow verification (e.g. if the break statement is used correctly in the code),
- Identifier uniqueness check.

As a result of this process, the previous AST is expanded with additional data collected during analysis. These data could be e.g. information about used and declared classes, attributes or methods, as it is implemented in the tool described in this paper.

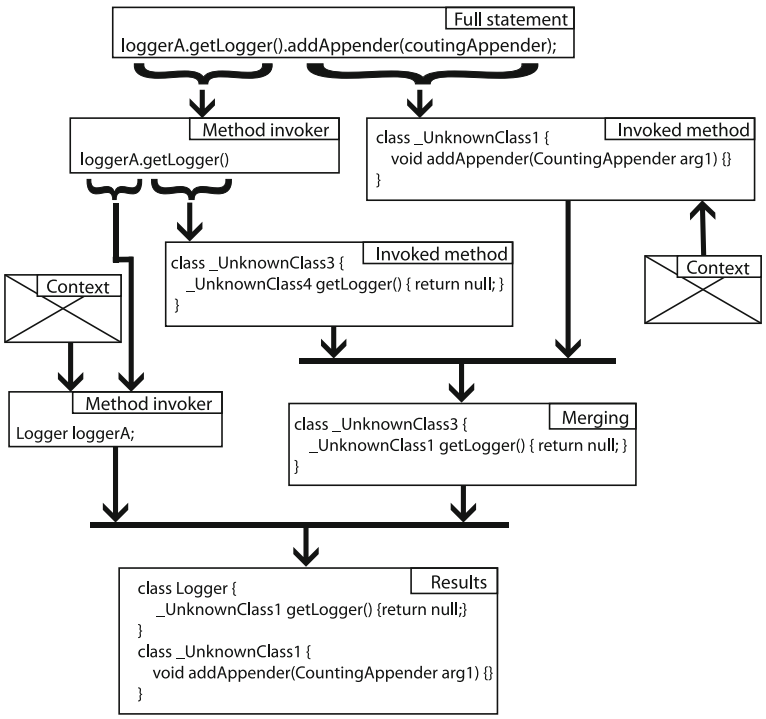


Fig. 3 An example of data extraction from the single statement implemented in the Log4J test method. The context block stores the information about the local variables which are already declared. Both context blocks represent the same instance of the storing object (the references to the same object)

An example of the single line analysis is presented in Fig. 3. Every statement is partitioned, and data are extracted from the parts. Parts are parsed in order from right to left.

Partitioning process is realized recursively till the moment when the parts become atomic statements, from which it is possible to obtain data, e.g. in the instruction from Fig. 3, the full statements can be divided into 2 parts: the method's invoker (`loggerA.getLogger()`) and the method's invocation (`addAppender(count-ingAppender)`).

The first part will be divided into the `loggerA` invoker (which is already an atomic identifier), and the `getLogger()` method invocation. During the data extraction specific for an atomic statement, we can obtain the following information by parsing this statement:

- method `addAppender(countingAppender)` has declared an argument, which has the same type as `countingAppender` variable,
- method `addAppender(countingAppender)` is called by the invoker part, so the method is declared in the invoker's class,
- there were no information about the return type, which implies that the method possibly does not return any value (`void`),
- the type of the invoker's statement is the type returned by `getLogger()` method, but there is no data about the name of this type,
- method `getLogger()` is invoked by the class instance, which is represented by the `LoggerA` local variable.

With this knowledge, and the information about the types of the mentioned local variables, we can specify definitions of used units (see Fig. 3).

The pattern of the data collecting realization differs between many types of the statements structure. However, the main structure of the statement decomposition is the same as the one described above—the first step is to decompose the instruction. The differences could be e.g. in the control instructions (like if–else) where the analyzer requires the Boolean statement as a condition (see Fig. 4), etc.

When the unit appears in more than one line of code, and the types required in the statements are different, a type conflict occurs. A problem of the conflicts during the analysis is solved using some simple rules:

- When both types are not known before (and will be generated as a result), the mutual base type will be created, and the unit's conflicted type will be converted

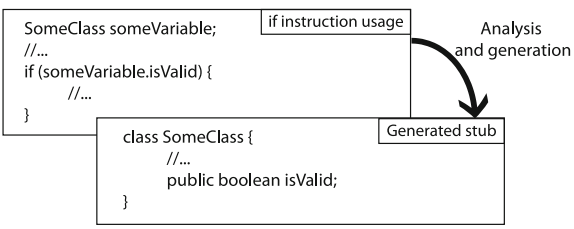


Fig. 4 Example of the attribute stub generated from the if statement

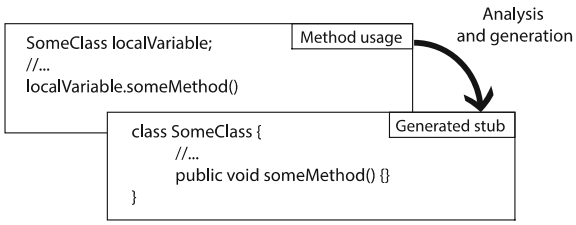


Fig. 5 An example of the method stub generated from the method invocation statement

to the base class. Previously detected classes will be defined as a child class of the new class,
- In the situation when one of the types is already defined, the unit will be adjusted to the defined type, and a new one will be declared as a child class of the existing one,
- When there is a conflict between two existing classes, which have the same parent, the unit will be placed in the parent class,
- In other cases, e.g. when both classes exist and they have different bases, the analysis is interrupted with an error (Fig. 5).

2.3 Existing Solutions

Currently, an *Integrated Development Environment* (IDE) like *Eclipse*, or *Intellij IDEA* offers the solutions for the missing units generation. They allow for choosing the package where the generated class will be located or for changing the suggested types.

Tools like these usually work in the context of the single instruction, with consideration of the already existing units. They do not provide the utility for the full code analysis. This approach is better if the user requires better control of the generation process, but it requires more time for realization.

The solution presented in this paper uses a holistic approach for code processing.

3 Tool Implementation

The application was designed to support the process of development in Java language. It is composed of two main modules:

- **Analyzer**—which obtains the data from the AST, and works only with the method with *JUnit's* `@Test` annotation,
- **Generator**—which converts the data to the source code.

```
JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
CompilationTask task = javac.getTask(null, fileManager,
      null, null, null,compilationUnits);
task.setProcessors(processors);
task.call();
```

Fig. 6 An example of the compiler process invocation

As a base for the project, the Java compiler (*javac*) was used. It allows to easily build the AST from the code. For further exploration of the syntax tree, the Java Compiler API is used, which is defined by the *JSR 199* specification [8]. Java Compiler API is a set of interfaces, which describe the utilities available from the Java compiler. The first version of the API was defined in Java 6 (Fig. 6).

The analyzer module works in two phases: the test method detection phase and the data extraction phase. The first one is realized by the extension of the `TreePathScanner` class defined in Java Compiler API.

The second phase module implements the strategy design pattern for specific statement handling. Each type has defined a class, which controls the process of statement partition and data acquiring.

Figure 7 presents the declaration of the method declared in the `StatementParserStrategy`, which is the base class for all statement handling class. The second and third arguments of the `parse` method allow for suggesting what type of the parser should be returned by the method (e.g. when the `if` statement is processed, the statement should return the `boolean` value) (Fig. 8).

The general algorithm of instruction analysis is defined as follows:

1. In case of the new code block (code nested between the curly brackets) the new context is created,
2. Statement partitioning,
3. Recursive analysis for each part (in this step the information about the types is obtained),
4. In case of the block end—returning to the previous context.

The context mentioned above is the structure for storing information about the local variables declared in the current block. Variables are visible only in the range of the block.

```
public Type parse(Tree tree, Type requestedType,
                  boolean forceNotNullResult);
```

Fig. 7 Declaration of the `StatementParserStrategy` class main method

Fig. 8 Main application screen

4 Case Study

As an example of code templates generation, a test class from the logging framework for Java—*Log4J* will be used.

In the study one of the methods defined in `LoggerTest` class will be analyzed and the results will be compared to actually implemented solution. The method will be tested separately (it will be excluded from the *Log4J* framework), which means that every unit used in the analyzed code have to be stubbed for successful project compilation.

Figure 9 presents the implementation of the method and the results of the generation. As we can see, the information about 3 new classes was obtained from the source method. One of them has an undefined name, because the analyzer's algorithm was not able to predict its type. The name of this class was not explicitly defined in any statement where it was used.

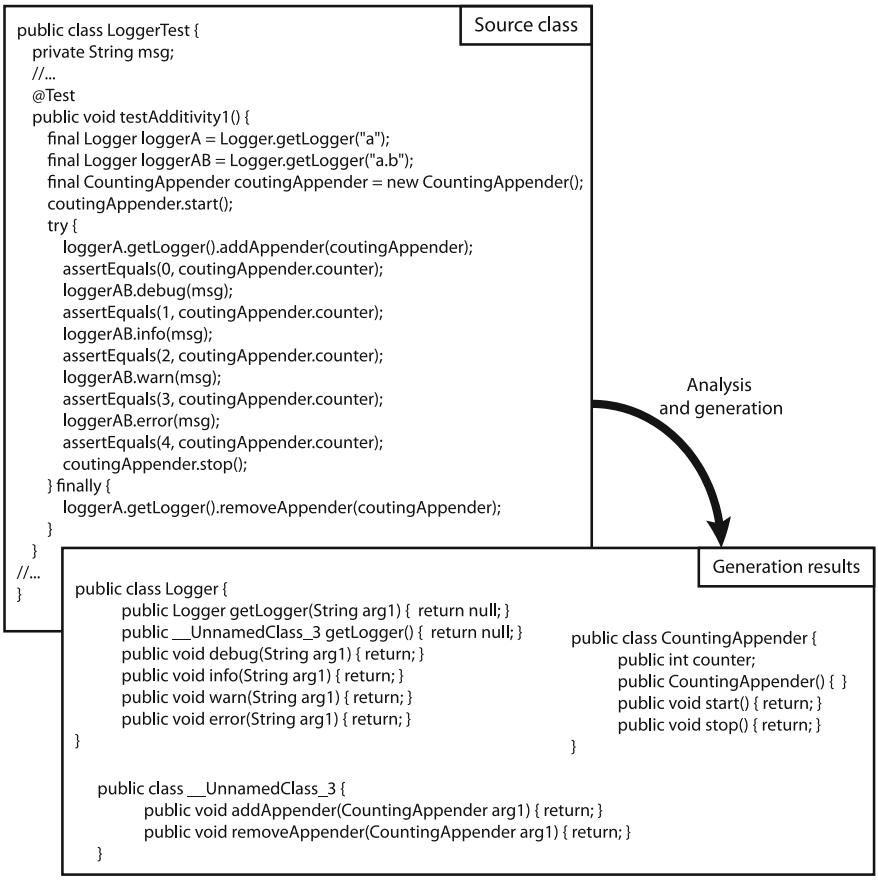


Fig. 9 Analyzed class and the results of the generation

4.1 Results Comparison

When we compare the code generated after the process with the real classes implementation, the results are acceptable. It allows to compile the code and to accomplish the red phase of RGR cycle. Additionally, the detected classes are similar to the real ones (implemented in *Log4J* framework).

However, there are some differences, like the real name of the `UnnamedClass_3`. In the framework implementation, the method `getLogger()` is defined in the `Logger's` base class. The return type of this method is defined as `org.apache.logging.log4j.core.Logger`, which is the class with the same name as the one detected during the analysis, but placed in a different java package. It was impossible to obtain this kind of data during the analysis, because none of identifiers has occurred in the test code.

5 Conclusions

The paper presents the concept of the Test-driven development (TDD) and a possible way for using the semantic analysis and code generation to support this methodology. A tool described in previous sections allows for reducing time required to pass the red phase of the RGR cycle by semi-automatic code stubs generation. The tool implements the concepts presented in the previous sections.

Described approach is semi-automatic, because in some situations it is impossible to identify the correct semantic from statements, and for the correct results it requires actions to take by the programmer. As an example, when some class has two attributes, which types are not possible to predict (but the programmer wants them to be declared with the same type), there will be generated two different classes for these. If the programmer wants to achieve his intent, he has to do it manually. The solution for the problem is to use annotations to control the generation process, which concept is mentioned later as a possible improvement.

In many IDE software tools for the stubs generation are used. The main difference between them and the solution presented in this paper is that it uses a holistic approach for code processing. In these utilities, the code generation is realized only in the local context, i.e. if we use these tools in the context of the statement, it will match the types only to become valid in this statement. In the solution presented here, the tool will try to match the unit types to be valid in all statements of the test method.

In the current state, the parser operates only on the syntactic and semantic analysis of some statement types. Currently, the tool does not handle all possible instructions of Java language. Even then, the results of the analysis and generation stubs for *Log4J* test allow for compiling the code. By the stubs generation, time required for passing the red phase is reduced. Thanks to that, the Test-driven development methodology becomes even more efficient.

There are many possible ways to improve the presented solution. Because the tool does not support the full grammar of the Java language, some instruction do not provide any information during the analysis. By increasing grammar coverage level, the tool will be able to obtain more data from the new statements. With this extension, the accuracy of the semantic analysis results will certainly increase.

The second idea for the tool improvement is to add special Java annotation handling. It will allow for controlling the process of semantic analysis by predefined annotations. For example, one of the possible ways of annotation usage is to disable some test methods from the process of parsing. In the current version of the tool, every method marked with *JUnit's* `@Test` annotation is processed. Additionally, the annotation can be used to suggest the type of some unit and can have an influence on further results.

References

1. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Addison-Wesley, Boston (2007)
2. Beck, K.: Test-Driven Development by Example. The Addison-Wesley Signature Series, Addison-Wesley, Boston (2003)
3. Earlay, J.: An efficient context-free parsing algorithm. Commun. ACM **13**(2), 94–102 (1970)
4. Gosling, J., Joy, B., Steele, G., Bracha, G., Buckley, A.: The Java Language Specification. Java SE 8 Edition. Oracle America, Inc. (2015), [http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf]
5. Grune, D., van Reeuwijk, K., Bal, H., Jacobs, C., Langendoen, K.: Modern Compiler Design. Springer, Berlin (2012)
6. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison Wesley, Boston (2006)
7. Shore, J., Warden, S.: The Art of Agile Development. O'Reilly, Beijing (2007). [http://my.safaribooksonline.com/9780596527679]
8. Sun Microsystems, Inc: JSR 199: Java Compiler API (2006). [https://jcp.org/en/jsr/detail?id=199]

Decomposition and Reduction of Indexing Structures with Use of the GPU Computations

Damian Raczyński and Włodzimierz Stanisławski

Abstract The purpose of this article is to determine the usefulness of the K-means method applied to decompose and reduce the TERM-BY-DOCUMENT matrix with use of the GPU computations. The reduction based on the truncated SVD decomposition is a long-term process for large matrices. The computational complexity of the SVD decomposition is $O(n^3)$. The use of the K-means decomposition before reduction should reduce the computational complexity. In addition, the use of the GPU should reduce the time of the whole process. In the article there is comparison of the times and the correctness of the results for the computing environments associated with the GPU.

Keywords LSI · K-means · GPU · Reduction · Parallel computing · SVD

1 Introduction

The modern LSI (Latent Semantic Indexing) methods, using the truncated SVD decomposition, are very popular for small homogenous collections of documents. For large non-homogenous collections of documents, computational complexity of methods based on the SVD decomposition prevents data reduction [1].

In the article we present the methodology for the application of the LSI method to a large collection of text documents. The TERM-BY-DOCUMENT matrix is decomposed, with use of K-means method, to set of matrices which have smaller sizes. The decomposed blocks are reduced with use of the truncated SVD decomposition. The presented methodology has been applied to reduce generally available set of TERM-BY-DOCUMENT matrices, containing the following documents: CACM, CISI, CRAN, MED.

D. Raczyński (✉) · W. Stanisławski
Państwowa Wyższa Szkoła Zawodowa w Nysie, Nysa, Poland
e-mail: damian.raczynski@pwsz.nysa.pl

W. Stanisławski
e-mail: wlodzimierz.stanislawski@pwsz.nysa.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_18

The environment, used in the experiment, related to GPU, are the programs developed in C++ language, with use the CUDA, CULA [2] and CULYA [3] libraries.

2 The LSI Method

The LSI method, introduced in [4], is a method for automatic text indexing. The retrieval systems based on the LSI are able to return a list of documents related in meaning to the words entered by the user, despite of the difference in the used keywords.

The LSI is a mathematical method, which is able to designating a simplified model based on spatial clustering of similar objects. The SVD decomposition is a crucial method, which provides the effectiveness of the LSI, however, the process of determining the decomposition is a major problem associated with the computational complexity. In the literature it is obvious, that the use of the original LSI method for large documents collection is practically not possible.

The proposed solution may consist of the performance of the TERM-BY-DOCUMENT matrix decomposition into smaller portions. Then, the reduction based on the LSI method is performed on the separated fragments. The advantage of this approach is possibility of independent simultaneous reduction of each element.

3 The K-Means Method

The K-means method belongs to the classification algorithms group. The clustering with use of the K-means has been known since 1967 [1]. This method is one of the simplest methods of grouping based on the unattended learning.

The algorithm is based on a grouping data into a certain number of clusters. Each cluster is associated with so-called centroid that represents the average value of all points belonging to the group. The centroids of all the clusters should be separated from each other by the maximum possible distance.

In the case of text documents, a document is assigned to that group which represents the centroid of the greatest similarity in relation to that document. In each iteration, the assignment of the document to the certain group can be changed. The process is repeated until centroids do not change their coordinates.

The K-means algorithm in MATLAB language is shown in Fig. 1. The `k_means` function takes 3 parameters—the TERM-BY-DOCUMENT matrix, the number of clusters and initial coordinates of the centroids. The function performs the following operations:

```
1.  function x=k_means(matrix, clusters, centroids)
2.    y=size(matrix,2);
3.    G_old=zeros(clusters,y);
4.    finish = 0;
5.    while(finish~=1)
6.      dist = zeros(clusters, y);
7.      for i=1:clusters
8.        for j=1:y
9.          dist(i,j)=sim(matrix(:,j), centroids(:,i));
10.       end
11.     end
12.     G=zeros(clusters,y);
13.     for i=1:y
14.       index_min=min_index(dist(:,i));
15.       G(index_min, i)=1;
16.     end
17.     if(max(max(abs(G-G_old)))==0)
18.       finish=1;
19.       x=G;
20.     else
21.       G_old=G;
22.       centroids = update_centroids(matrix, G);
23.     end
24.   end
25. end
```

Fig. 1 The source code of the K-means algorithm in MATLAB

LINES: OPERATIONS:

7–11 Determining the similarity of documents' vectors with the centroids. The measure of similarity used in the article is the cosine distance

13–16 The shortest distance vector-centroid is determined with use of the `min_index` function. The index of the nearest centroid is used to create the binary matrix of connections `G`. The formed matrix determines which document belongs to which group (which vector is assigned to which centroid)

17–19 Checking the condition whether any vector of document changed group with compare to the previous iteration. If the change does not occur, the algorithm is finished and the result is the last G matrix

20–22 In case, where a change has occurred, the function update_centroids is invoked which updates the coordinates of the centroids.

The K-means algorithm is very sensitive to the initial selection of the centroids' coordinates. In order to determine the final solution, it should be executed many

times with different initial coordinates. A common method used to determine the initial centroids is a random selection a certain number of vectors from the TERM-BY-DOCUMENT matrix. The K-means does not determine the optimal number of groups, which will be divided into a set of documents. Their number is determined arbitrarily before the start of the algorithm. In order to determine the optimal number of clusters in a given set of documents, the algorithm should be performed repeatedly for different values of this parameter [5].

4 Decomposition and Reduction

A large collection of documents can be divided into a number of smaller collections, each of which contains documents related to a certain subject. Reduction with use of the SVD decomposition is characterized by high computational complexity ($O(n^3)$). The decomposition of the TERM-BY-DOCUMENT matrix into m blocks, assuming equal size of divided blocks, reduces the computational complexity to a level of $O\left(m(n/m)^3\right)$. In addition, the decomposition of the matrix into a number of parts allows the simultaneous reduction with the use of the cluster system. Each block can be a separate reduction task performed on a separate cluster node.

For the original TERM-BY-DOCUMENT matrix (1)

$$A = [a_1, a_2, a_3, \ldots, a_n] \tag{1}$$

where $\mathbf{a}_i$ denotes document's vectors, decomposition into p parts causes the secretion of blocks (of smaller size than the original matrix), satisfying dependences (2) [1].

$$\begin{gathered} \bigcup_{i=1}^{p} \alpha_i = \{a_1, a_2, a_3, \ldots, a_n\} \\ \alpha_i \cap \alpha_j = \emptyset \text{ if } \quad i \neq j \end{gathered} \tag{2}$$

After the decomposition, the original order of the vectors in the TERM-BY-DOCUMENT matrix changes (3)

$$A' = [\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_p] = [a'_1, a'_2, a'_3, \ldots, a'_n] \tag{3}$$

for each of the blocks the corresponding centroid is obtained (4) [1].

$$C = [c_1, c_2, c_3, \ldots, c_p] \tag{4}$$

For a given query vector it is possible to determine the nearest cluster by comparing the measure of the similarity between the query vector and the centroid vectors. If only a part of the specified clusters are taken into account when

searching for documents that match a user's query, it causes the speed up of the searching process.

Gao and Zhang in [1] present three main strategies available for use in a retrieval system.

- NC+SVD (Nonclustered retrieval)—operations are performed on the TERM-BY-DOCUMENT matrix, which has been only reduced with the use of the SVD method.
- FC+SVD (Full clustered retrieval)—after decomposition, with the use of K-means algorithm, the obtained blocks are reduced. The search for the most matching documents is performed by determining the similarity measure of the reduced query vector with the all reduced documents' vectors.
- PC+SVD (Partial clustered retrieval)—decomposition and reduction are performed in the same manner as in the FC+SVD strategy. Searching the most matched documents in this case is based on comparing the reduced query vector with the centroids belonging to the reduced blocks. In this way, the blocks which are entirely dissimilar to the query are ignored. The set of clusters, in which the search is performed, is less than the number of all clusters.

5 Experiment

The K-means algorithm takes the initial parameter which is a collection of centroids. In this article, the initial centroids are randomly selected set of vectors from the TERM-BY-DOCUMENT matrix. This way of selection centroids entail two significant drawbacks. The first one is related to the possibility of selecting the same vector several times. The second one is associated with the possibility of determine the vectors which are close to each other. The value of the cosine similarity varies form -1 (no similarity) to 1 (a hundred percent similarity). In the case of the method used in the article, the initial centroids differ from each other by at least 0.1 (in the sense of the cosine distance).

The algorithm has been developed for the GPU environment with use of the CUDA, CULA and CULYA libraries. Due to the fact that the method is a heuristic algorithm, subsequent calls may return different solutions. The size of each part of the decomposed matrix is not constant on the subsequent algorithm calls (the heuristic is associated with the random initial centroids).

The structures of the TERM-BY-DOCUMENT matrix before and after the decomposition, for selected cases, are presented in the Figs. 2, 3, 4 and 5.

In the Figs. 6, 7, 8, 9, 10 and 11, there are presented sizes of decomposed parts of the TERM-BY-DOCUMENT matrix, depending on the considered reduction case (for calculation on single precision).

In the Fig. 12, there is presented the mean square error of reduction, expressed by the formula (5).

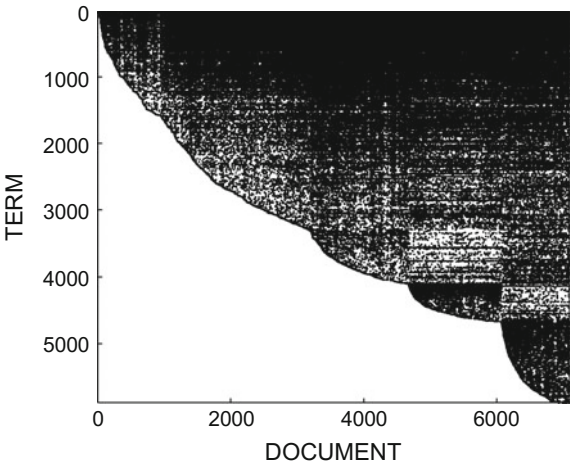


Fig. 2 Matrix structure for n = 1

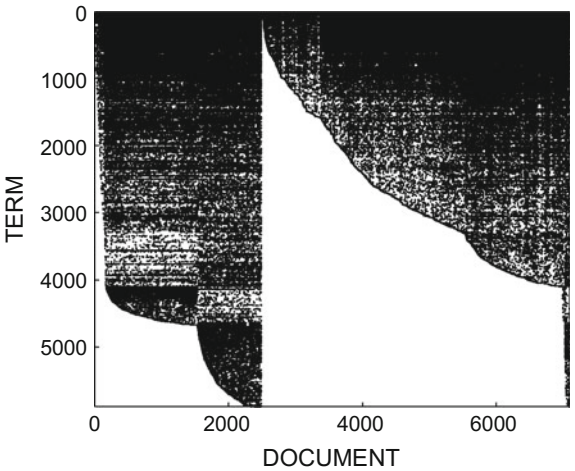


Fig. 3 Matrix structure for n = 2

$$MSE = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(COS\_ORG_{ij} - COS\_RED_{ij}\right)^2 \quad (5)$$

where: n = 7095, COS_ORG_{ij}—the cosine distance between the i'th and j'th documents in original TERM-BY-DOCUMENT matrix, COS_RED_{ij}—the cosine distance between the i'th and j'th documents in reduced TERM-BY-DOCUMENT matrix.

The final evaluation of the used methodology has been determined in the following way. For the 100 randomly generated user's query vectors, the most matched document are determined from the original TERM-BY-DOCUMENTS

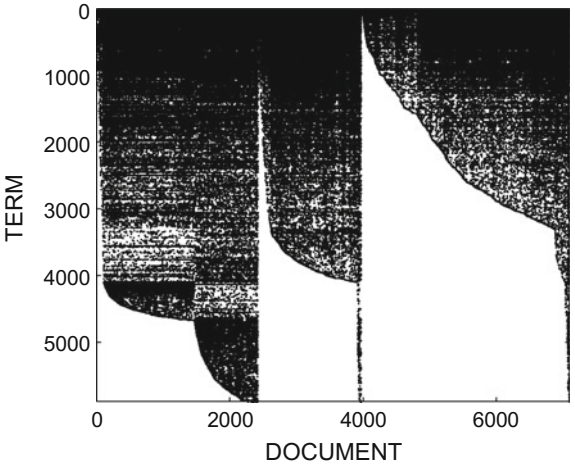


Fig. 4 Matrix structure for n = 3

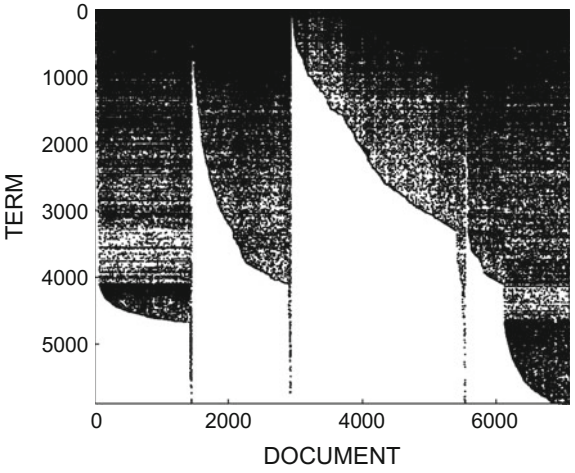


Fig. 5 Matrix structure for n = 4

matrix. Then, for the same query vector, **p** (where p = 1, 10, 20, 30 …, 100) the most matched documents in the reduced TERM-BY-DOCUMENT matrix are determined. The precision is computed as a percentage of the same documents in both collections. The obtained results are shown in Figs. 13, 14, 15, 16, 17 and 18.

The average time required to reduce the TERM-BY-DOCUMENT matrix, depending on the number of decomposed parts, is shown in Fig. 19. Each column shows the average data for the reduction to 1, 2… 20, 100, 200,… 1000 rows, for single precision calculations.

In a particular case of reduction, the reduction of the computation time with the number of blocks cannot be assumed (due to the heuristic K-means). The overall trend, however, is visible. The minimum and the maximum reduction time, for the particular case, is shown in Fig. 20.

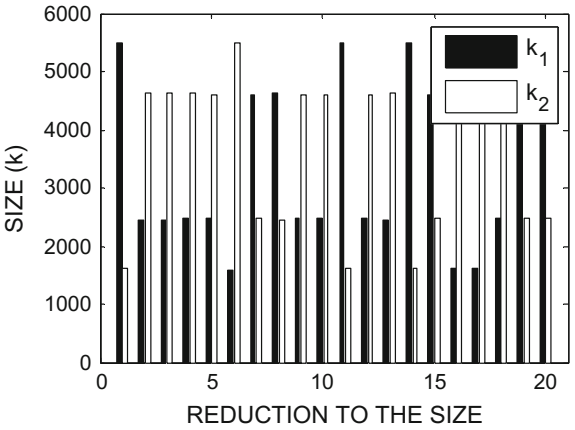


Fig. 6 Reduction to 1–20 rows, n = 2

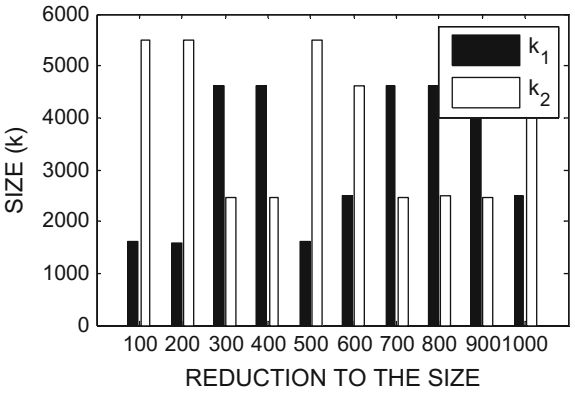


Fig. 7 Reduction to 100–1000 rows, n = 2

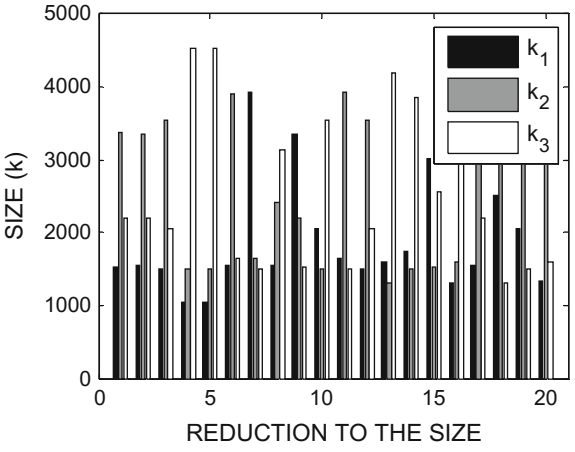


Fig. 8 Reduction to 1–20 rows, n = 3

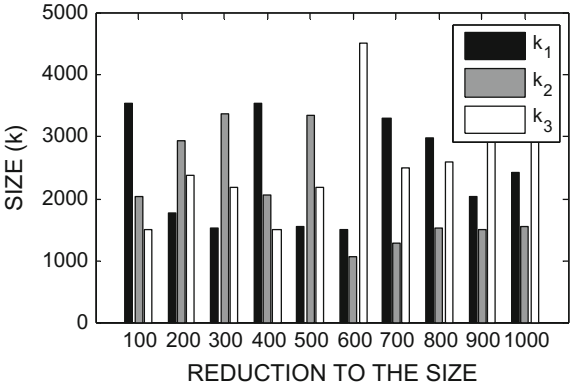


Fig. 9 Reduction to 100–1000 rows, n = 3

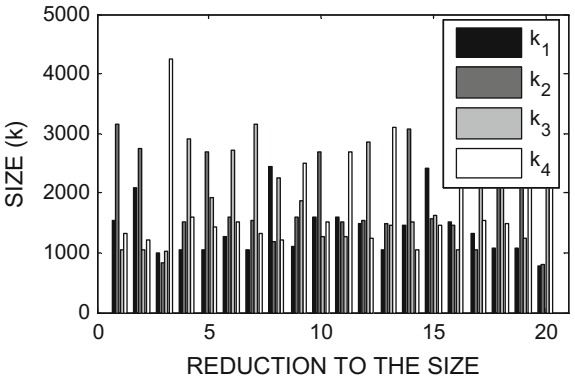


Fig. 10 Reduction to 1–20 rows, n = 3

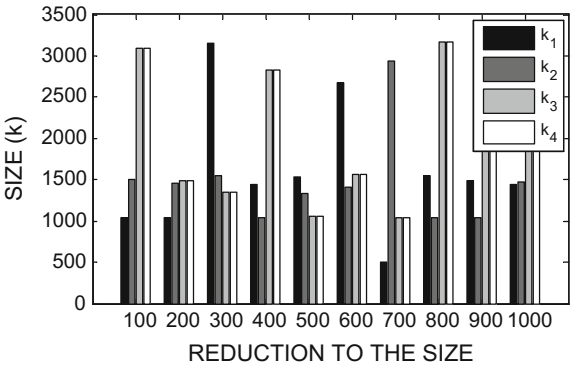


Fig. 11 Reduction to 100–1000 rows, n = 4

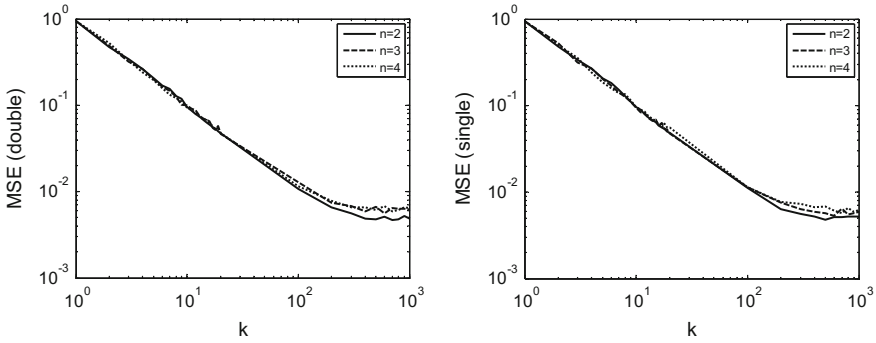


Fig. 12 The MSE errors of reduction

Fig. 13 n = 2 (double)

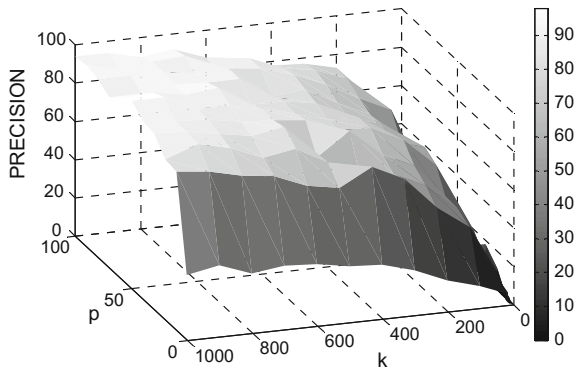
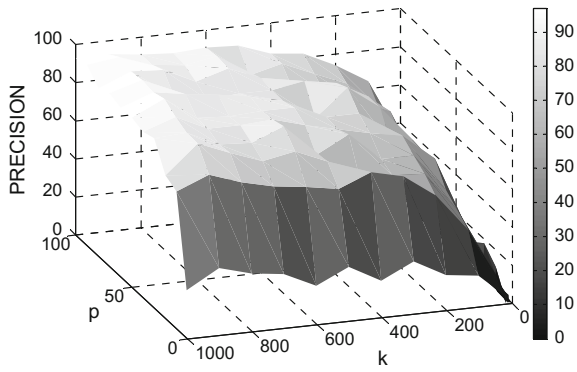


Fig. 14 n = 3 (double)



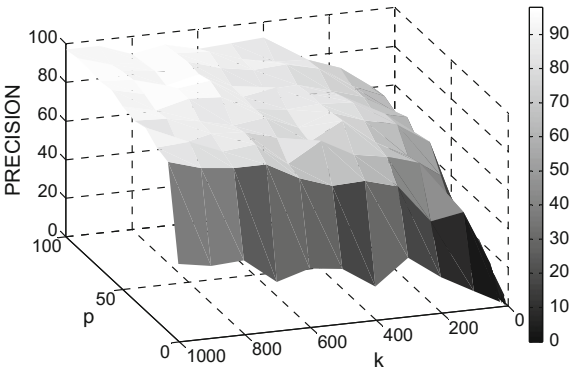


Fig. 15 n = 4 (double)

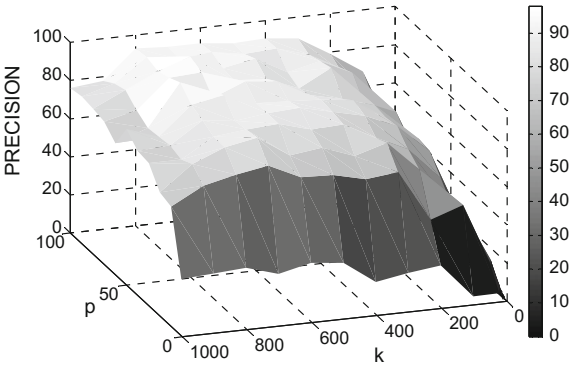


Fig. 16 n = 2 (single)

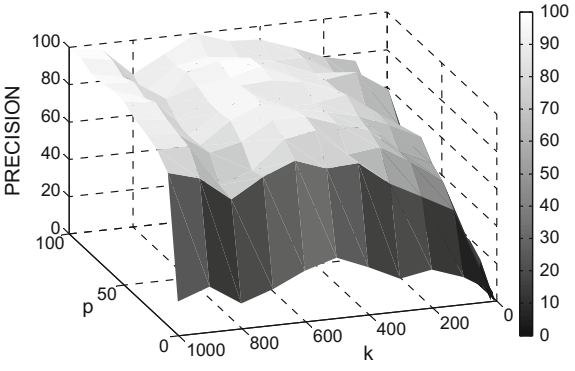


Fig. 17 n = 3 (single)

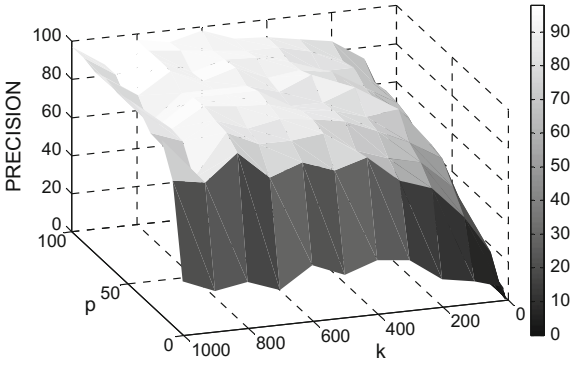


Fig. 18 n = 4 (single)

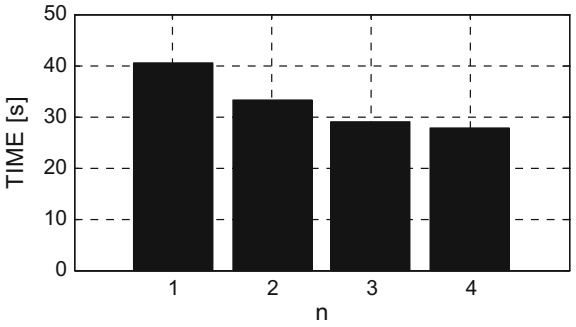


Fig. 19 The average time required to reduce the TERM-BY-DOCUMENT matrix, depending on the number of decomposed parts

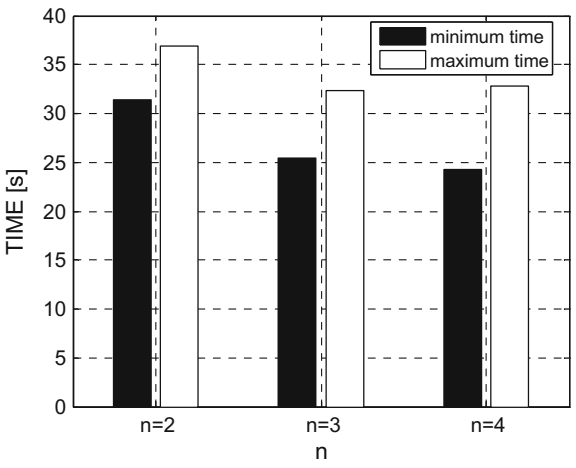


Fig. 20 The minimum and the maximum reduction time

6 Conclusions

The use of the K-means algorithm, in order to decompose the TERM-BY-DOCUMENT matrix, reduces the time needed for reduction. In a particular case, taking into consideration the minimal reduction times, the use of decomposition allowed reduce the reduction time by almost half (to 62 %). As also shown in the article, the GPU computing environment is suitable for the reduction of the TERM-BY-DOCUMENT matrices.

The main drawback of the presented methodology is unpredictability of the K-means algorithm. Should be remembered that the greatest reduction in computational complexity is achieved when the algorithm determines blocks which having a similar sizes. When the decomposed blocks vary much in size, the whole process will give a small reduction in computational complexity.

References

1. Gao, J., Zhang, J.: Clustered SVD strategies in latent semantic indexing. Inf. Process. Manage. **41**(5), 1051–1063 (2005)
2. Raczyński, D.: Matrix computations using GPU. In: Information Systems Architecture and Technology. Contemporary Approaches to Design and Evalutionary of Information Systems, pp. 39–48. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław (2014)
3. Raczyński, D., Stanisławski, W.: Controllability and observability gramians parallel computation using GPU. J. Theor. Appl. Comput. Sci. **6**(1), 47–66 (2012)
4. Deerwester, S., Dumais, S.T., Furnas, G.W., Landeuer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
5. MacQueen, B.J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkley Symposium on Mathematical Statistical and Probability, pp. 281–297. University of California Press (1967)

Social-Media Data Analysis Using Tessera Framework in the Hadoop Cluster Environment

Martin Sarnovsky, Peter Butka and Jakub Paulina

Abstract The presented paper describes the design and implementation of R functions for twitter feeds analysis and visualization based on a combination of analytical technologies with big data processing tools. The main idea was to utilize the Hadoop processing framework and its storage and computational capabilities in analytical tasks designed and implemented in R language. For such purposes, we decided to use the Hadoop HDFS and MapReduce v2 for storage and handling of the processing logic connected via Tessera framework to analytical functions written in R. The results of the analysis were presented as the graph visualizations. Visualizations were implemented using the Trelliscope framework for flexible visualizations of large complex data in R environment in fast and effective fashion.

Keywords Hadoop · Mapreduce · Distributed computing · Data analysis

1 Introduction

The volume of the data available in various databases is significant. The rate how the data volume is increasing and greatly exceeds the rate computational processing power growth of the computers. One of the main challenges of data processing nowadays is except their storage (using the distributed filesystems and databases) also distributed data processing and analysis. Problems with big datasets include

M. Sarnovsky (✉) · P. Butka · J. Paulina
Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Letna 9/A, 04200 Kosice, Slovakia
e-mail: martin.sarnovsky@tuke.sk

P. Butka
e-mail: peter.butka@tuke.sk

J. Paulina
e-mail: jakub.paulina@tuke.sk

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9_19

also its recording, storage, searching, and analysis. In addition, each analysis produces another new data.

To process and analyze the big data it is necessary to use the computing tools capable of fast and efficient handling of the previously mentioned tasks. Without designing of the new models and using suitable software and hardware resources, a solution of the majority of these tasks would not be possible. Usually, these tasks involve the distributed data and distributed processing. One of the most commonly used computation models in that area is MapReduce [1], aimed at parallel processing of the big distributed datasets and was successfully used in various applications. Using this computational model, problem solution is divided into the two steps—Map and Reduce [2]. Each step is executed in parallel and distributed fashion on multiple computing nodes. Its main advantage is (besides speed) a fact, that the distribution logic, load balancing, fault tolerance and similar issues are handled by the framework itself and not by the developer. MapReduce moves the processing to the data. Its most familiar open source implementation is Apache Hadoop. Limitations of Hadoop include the problems with iterative tasks, where the most of the machine learning tasks belong [3]. Several limitations were removed in MapReduce v2 implementation, which used YARN (Yet another resource negotiator) as the resource manager [4]. The limitations also lead to alternative solutions including various in-memory frameworks such as GridGain or Spark. Apache Spark is 100-times faster in memory and 10-times faster od disk than Hadoop, which makes it an ideal tool for big data processing tasks since it supports cyclic data flows and in-memory computation [5].

Based on these platforms, numerous big data processing, and analytical tools were developed for example Impala, Hive, HBase, Stratosphere, etc., which extend the Hadoop platform with database functionalities. Another set of tools are aimed at machine learning, such as Mahout, which is basically a distributed alternative based on Hadoop to already existing standard machine learning and statistical analysis tools such as Weka or R. Besides Mahout, some other approaches exist, for example MLlib as the similar platform on the Spark platform, Radoop, which enables to work with the big data in the popular knowledge discovery tool RapidMiner, where Hadoop cluster serves as a storage capacity and to perform the computations.

On the other hand, several tools enabling connection of R to Hadoop clusters are available. RHadoop provides the MapReduce interface and Hadoop distributed filesystem to analytical applications developed in the R language [6]. RHadoop consists of several R packages providing Hadoop stack functionality to the R such as rhdfs (HDFS access from R), rhbase (HBase connector to R), rmr2 (MapReduce v2 operations). Another one is Tessera, open source environment for analysis of large datasets, which will be described in more detail in the next chapter.

2 Tessera

In this paper, we describe the system implemented and tested on selected cloud-based infrastructure with the usage of Tessera[1] software, which provides an open-source computing environment for analysis of large complex data. The main advantage of the environment is the combination of Big Data technologies, specific visualization framework for the complex exploration of data, and connection to R[2] statistical tool (and its packages).

Tessera provides an opportunity for the data analyst to code his/her analysis directly in R, while in the backend a distributed parallel computational environment is used (like Apache Hadoop). The way how the implementation of analysis is realized in Tessera is called Divide&Recombine (D&R) [7]. While the basic idea is similar to MapReduce technique, the details show more differences where D&R goes further in analytical tasks. First, the analyst will divide the data into subsets and writes them to disk (so-called *S* computation), series of analytical methods is then applied to subsets independently (*W* computations). The analytical methods are statistical methods with categorical or numeric output or visualization methods with visual output. No communication is used during this step between analytical threads used for subsets. Then recombine is applied (*B* computations) in order to achieve combined view of result for particular type output. The recombination methods can be of different types and granularity. Types are statistical recombination (the result is a value-based combination from complex data), visual recombination (combination of visuals on data with different granularity—we can see summarized visualization or detailed data view), or analytic recombination (which provides new data subset for other analysis).

In order to provide D&R Tessera contains three main components:

- datadr[3]—provides an interface to D&R operations, with the possibility to connect it to different distributed computing technology (currently supports in-memory, local disk, multicore, Hadoop, some preliminary version for Apache Spark). Programming is available in R and data is represented as R objects. In datadr are two basic data types—distributed data frames (DDF) and distributed data objects (DDO). DDF can be described as data frame divided into small parts, each of them contains a subset of rows and can be in the different cluster node. DDO is more general representation where each subset can be an object with arbitrary structure. Every DDF is also DDO.
- RHIPE (R and Hadoop Integrated Programming Environment)[4]—this package allows programmers to run Hadoop MapReduce jobs directly from R. The package is used by datadr if Hadoop is used for the backend.

[1]Tessera—http://tessera.io/.

[2]R project—https://www.r-project.org/.

[3]Datadr—http://tessera.io/docs-datadr/.

[4]RHIPE—http://tessera.io/docs-RHIPE/.

- Trelliscope[5] [8]—provides flexible visualization of large complex data in selected detail from R environment. The main advantage is in the scalable division of data to small parts and application of plot functions, arrange particular plots to grid with the possibility to interactively sort, filter, query them for specific information. Trelliscope supports multipanel displays with a large number of subsets and views them interactively and meaningfully. An important feature of Trelliscope is the organization of its displays and other visual artifacts into so-called visualization database, which can be easily shared by other analysts.

Regarding the data flow within the Tessera environment, we have to say that data structure which persists all DDO and DDF objects are matched as key-value pairs. For our purposes the key is usually identificator for specific subset and value is the subset of data corresponding to this key. Therefore, DDO and DDF is actually a list, where every element contains a key-value pair. Another important feature of data management is in persistent storage for datasets. The specific case is when larger datasets could not be stored in memory and should be stored to disk (localDiskConn function) or HDFS (function hdfsConn) using backend connection.

Finally, datadr provides functions for data operations which can be applied to datasets. These operations can be divided into three parts: (1) Data operations based on specific subsets (based on MapReduce computations, e.g., implemented functions like divide, recombine, drlapply, drfilter, drjoin, etc.), (2) Data operations independent from specific subsets (quantile and aggregation functions), (3) Read data operations (extended functions for import of datasets). Trelliscope functions are related to visualization aspects and include the connection to visualization database, display functions (makeDisplay, view, prepanel) and their inputs, panel functions and a specific type called cognostic functions—these are applied on each subset and returns list included into dataframe (contains any useful information for later display).

3 Design of the Proposed System

The following chapter will be focused on the underlying infrastructure used in described system and how it was utilized in the proposed solution.

3.1 Hadoop Cluster Infrastructure

Big data processing requires computational resources capable of handling and storage of big data. In our solution, we used a small-sized cluster consisting of a

[5]Trelliscope—http://tessera.io/docs-trelliscope/.

master node and three worker nodes. Master is equipped with 64 GB RAM and contains 8 CPU cores, workers are equipped with 32 GB RAM and 4 CPU cores. Cluster nodes operate CentOS operating system and run Cloudera Hadoop distribution in version 5.6.0. Hadoop cluster uses HDFS (Hadoop distributed file system) as a file storage and uses YARN (Yet another resource negotiator, or MapReduce v. 2.0) as a task processing paradigm. Hadoop cluster was also configured with several other related tools and packages such as HBase, Hive, Pig, Hue GUI, but those were not directly used in proposed work. Cluster configuration had to be upgraded to support distributed processing of R scripts. For that purposes, on each node R and R packages related to distributed processing (described in Chap. 2) were configured. Master was also extended witch Rstudio and Shiny server.

3.2 *Architecture of Proposed Solution*

Figure 1 describes the architecture and data flow of proposed solution. HDFS is used as a data storage, contains input data and stores the results. Data in HDFS are divided into the blocks and replicated across the cluster (replication factor set to default value—3). Implemented R functions are applied on distributed data using datadr. Datadr sends a request to the Rhipe package. Rhipe loads user-defined job settings and specifies Hadoop job settings according to them. YARN allocates the available resources according to those settings, locates the data blocks stored in HDFS and starts the map tasks of the whole job. Intermediate results produced by the mappers are stored into the cache or HDFS. Mappers perform combine method (local aggregation) after mappers are finished. After all map tasks are finished, YARN executes reduce phase, which aggregates the results of the whole job. Results are stored in HDFS, YARN sends job execution confirmation to Rhipe and output data are visualized by Trelliscope.

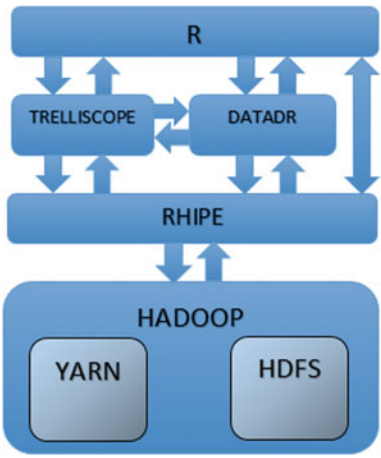


Fig. 1 Architecture of the system

3.3 Design of the Visualizations

Visualization is handled by the Trelliscope package, which the main purpose is to provide the graph visualizations in fast and effective fashion. We decided to implement those visualizations:

- visualization of tweets counts in New York area using *Rbokeh* and *leaflet* library —the goal of the visualization is to show the tweets daily frequency in hourly intervals for different languages, our aim was to use two different libraries for the same task.
- visualization of the popularity of most frequently used Twitter accounts—in this visualization, the goal was to extract the most frequently used accounts, combined them with the location of the tweets and ordering them according to the favorites and followers count.
- visualization of most popular hashtags—the goal of this visualization is to extract the most popular hashtags in particular language and location and visualize them in bubble-chart style.

4 Implementation of Visualizations and Experiments

In this section, we describe selected dataset based data from the Twitter network, necessary preprocessing steps, as well as implementation and experiments with the proposed visualizations.

4.1 Description of Input Data and Preprocessing

For our experiments, we have decided to select Twitter network data, which is a microblogging network for exchange and sharing of the short public messages called tweets. This messages are limited to 140 characters and may contain links to blogs, web pages, pictures, videos, etc. Using the follower functionality, it is possible to create a data source with some specific interest. The Twitter network connects more than 200 million users and provides hundred millions of contributions per day.

Selected data were extracted using data stream platform developed within the UrbanSensing project,[6] for our purposes we used some subpart of data from New York (more than 60,000 tweets). In order to clean the data, we had to implement

[6]UrbanSensing project—http://urban-sensing.eu/.

and apply some preprocessing filters: TimeSeparator (identifies and separates date/time inputs), ListFixer (extracts important attributes included in encapsulated lists), TimeHourMinSeparator (extracts exact time attributes), dateMaker (extracts exact date attribute in the correct format).

Preprocessing steps were also implemented within Tessera environment and applied using Hadoop jobs on dataset divided into 10 subparts, which helps to reduce the time for preprocessing by more than 5 times.

After preprocessing step, Twitter data were stored in distributed data frames based on two aspects. First, data were divided and cleaned to have particular tweets, which contains attributes presented in Table 1. Then, data were also prepared in a form representing data on users, which contains attributes presented in Table 2.

Then, data were also prepared in a form representing data on users, which contains attributes presented in Table 2.

Table 1 Attributes extracted by preprocessing steps for particular tweets

Attribute	Type	Description
id	numeric	Unique identification of user
lang	factor	Identifies language of tweet (“und” if not identified)
text	character	Text of tweet
place	character	Place where tweet was created
day	factor	Day of creation date from {“Mon”, Tue, …, “Sat”, “Sun”}
month	factor	Month of creation from {“Jan”, “Feb”, …, “Nov”, “Dec”}
dayNumber	numeric	Day of creation as number in particular month
time	character	Time of tweet within 24 h day
year	numeric	Numeric representation of year of tweet’s creation
created	date	Full date in Date format in R (unseparated)
coordLong	numeric	Place of creation—coordinates (longitude)
coordLat	numeric	Place of creation—coordinates (latitude)
hashtagText	list	List of texts of hashtags used in tweet
media_url	list	List of links to different media (pictures, videos, etc.)
media_type	list	List of type of media in tweet

Table 2 Attributes extracted by preprocessing steps for particular users

Attribute	Type	Description
id	numeric	Unique identification of user
screen_name	character	Displayed name of user
followers_count	numeric	Number of user’s followers
favourites_count	numeric	Number of user’s favourite tweets
statuses_count	Numeric	Number of user’s used statuses

5 Implementation of Selected Visualizations and Experiments

Finally, we provide the details on particular visualizations and example of the experiments with them according to processed dataset with tweets from New York and New Jersey area.

Application for the visualization of tweets counts using different libraries

As a first experiment, we decided to count tweets from different perspectives. First, we identified tweets written in different languages, as well as divided data according to places where tweets were created. Thanks to Tessera functions within flexible D&R, preprocessed frames can be reused for other recombination od data for new evaluation. As visualization result, we were able to provide tweets counts according to languages, places, time, in any combination. Moreover, the main advantage of Trelliscope is the possibility to use functionality for visualization of many subparts of data with backend prepared for a fast and effective change of visualized data and perspective. Therefore, the user is then ready to search in data, but also change panel layouts (setup number of rows/columns for subparts), select properties according to which subparts are shown (filter by properties of data), show global statistics using univariate or bivariate filters, sort panel tables. An example of tweets count visualization per place of creation is shown in Fig. 2.

Due to the fact that Trelliscope supports the usage of any available visualization library for particular graphs, we have also decided to use other opportunities. First,

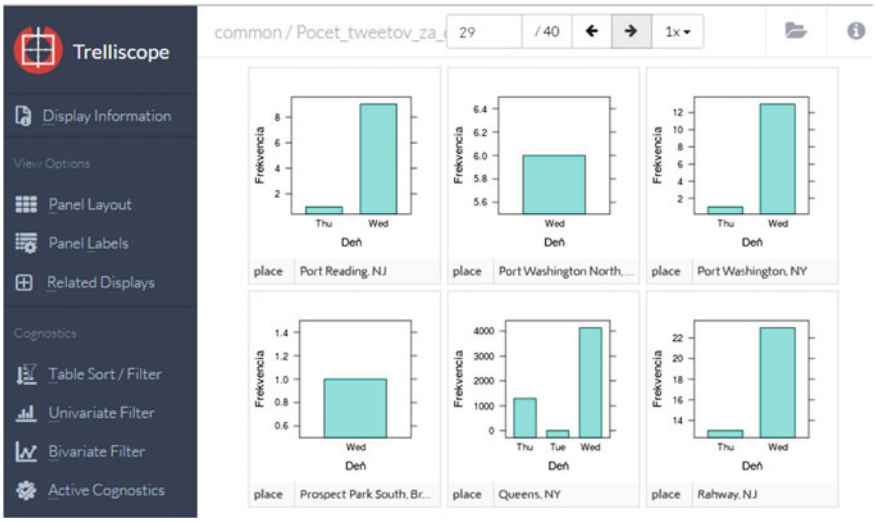


Fig. 2 Visualization of tweets counts per places using Trelliscope

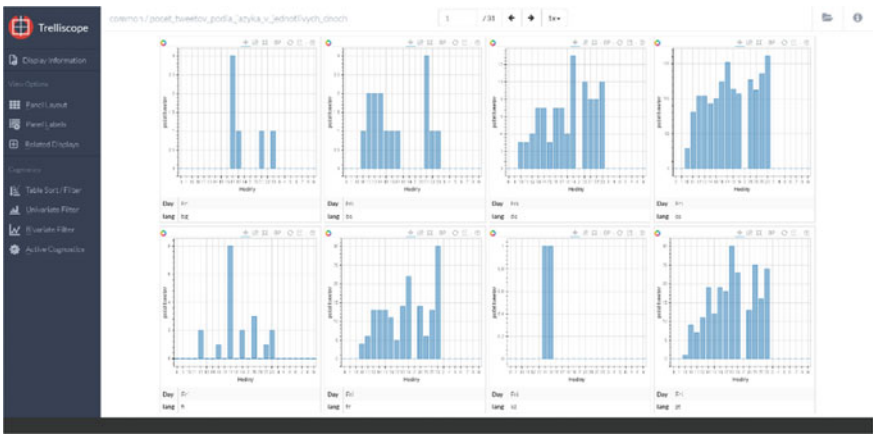


Fig. 3 Visualization of tweets counts per day, language and time, particular graphs are visualized using graph type from *bokeh* library in Trelliscope environment

we tried to reuse *bokeh* library,[7] which has its interface to R through a package called *Rbokeh*.[8] In this we provide interesting visualization able to show the analyst (user) tweets counts during the time in day in particular graphs (realized using *Rbokeh* chart type), where each graph (subpart) was identified by day and language, i.e., we have aggregated counts into three dimensions—particular day (like Wednesday), time (hours), and language. An example of the application interface is shown in Fig. 3.

The last application we have prepared for visualization of tweets counts was specific reuse of library for maps. In this case we decided to reuse *leaflet* library.[9] One disadvantage of this visualization was in fact, that many points can be shown in the map for tweets, thanks to leaflet feature it is possible to aggregate also dots in some area (depending on current zoom) and show a larger circle with a number. Hence, we have prepared leaflet based visualization with subparts as maps containing the counts of tweets in particular map per day and language (and according to zoom), where circles of different color show the counts for particular part of current maps zoom (blue icon is one tweet if aggregation is not needed). An example is shown in Fig. 4.

Application for the visualization of popularity of most frequently used accounts

For another visualization, we have used preprocessed data and extracts information on users, with followers and favorites counts. Pre-analysis of users shows

[7] http://bokeh.pydata.org/.

[8] http://hafen.github.io/rbokeh/.

[9] http://leafletjs.com/.

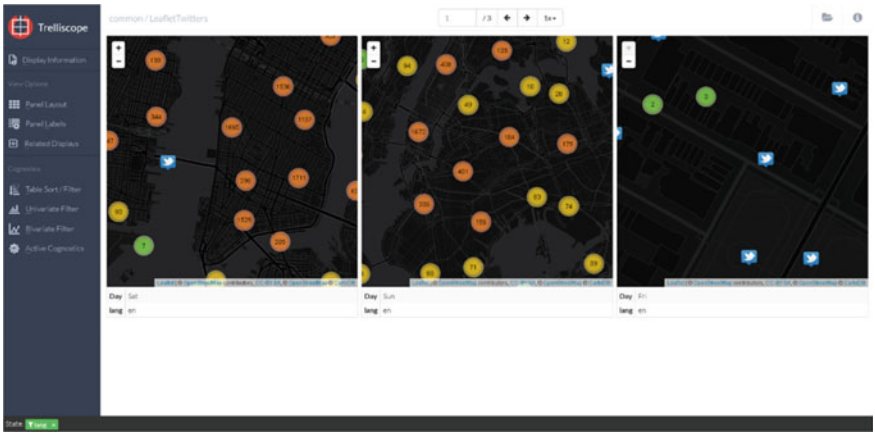


Fig. 4 Visualization of tweets counts per language and day, with maps (based on *leaflet* library) where the place of tweets creation is shown (numbers are aggregated numbers of tweets for smaller zoom, *blue* icon are particular tweets if zoom is large). Here, we can see places of tweets during the weekend in English (at parts of Manhattan)

that favourites_count has larger "weight" (1.5× larger count in average). Therefore, we decided to create a new attribute for this visualization called the *score*, which is counted as summary favourites_count and 1.5*followers_count. While Trelliscope has no problem to show many graphs for a large number of users, it is more interesting to show users which have at least some number of tweets in particular days. Here we decided to filter out users with less than 10 tweets. Then we prepared visualization of score according to the user. As an interesting feature, we also added through cognostic function also a most frequent place of creation of tweet, link to Wikipedia page related to this place, (user can also add other). The example of a particular user popularity (score) graph in time (with placed tweets in time and their popularity) is shown in Fig. 5.

Similarly, as in the previous case, Trelliscope environment can be used for setup of visualization of graphs on users and their popularity changes, including the selection of what should be on axis, graph, etc. An example of Trelliscope interface with more graphs on the popularity of users is shown in Fig. 6.

Application for the visualization of most popular hashtag words

For the last visualization, we decided to prepare an application which shows most popular hashtag words in data based on the bubble visual technique (bubbles with different size have a different value of the variable, bigger bubbles with larger value are in the middle of a visual display).

Of course, similar to bubble technique wordcloud can be used. For the purpose of this visualization, we have decided to show mostly used hashtags for particular languages. Figure 7 shows the example of bubble diagram of most popular hashtags in New York data for Korean and English language.

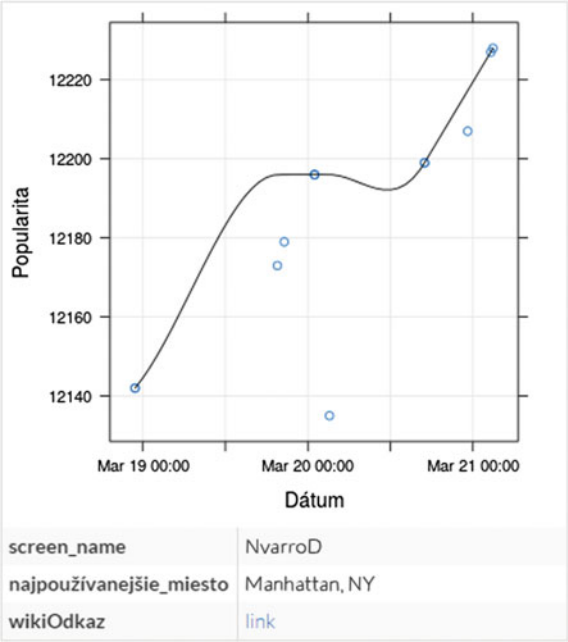


Fig. 5 Visualization of user's popularity (score) in time (with tweets as *blue circles*) based on favorites and followers counts, including the name, most common place from which tweets were sent, or link to the wiki page for this place

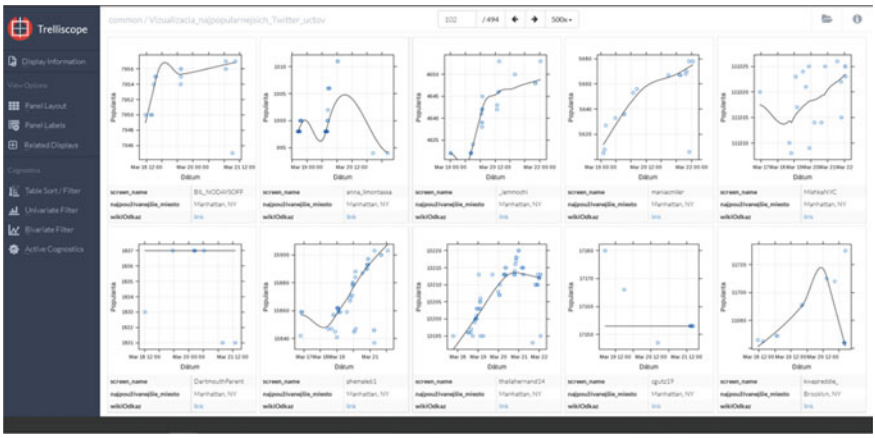


Fig. 6 Visualization of popularity (score) of more users in time using Trelliscope

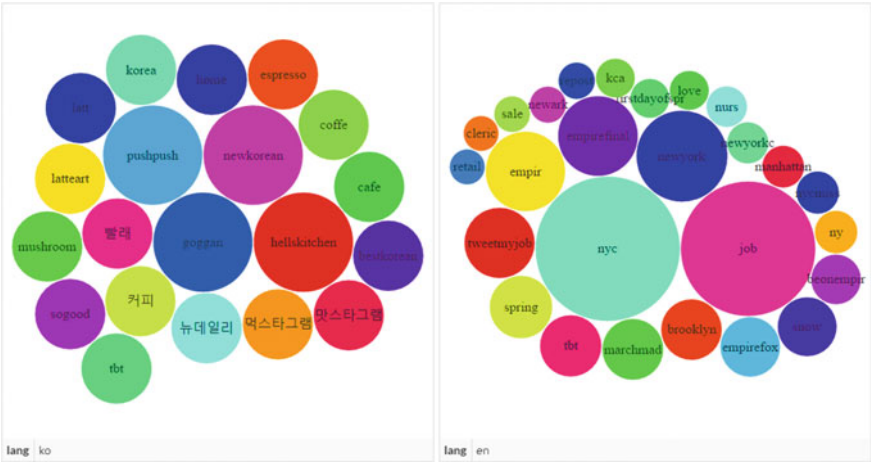


Fig. 7 Visualization of popular hashtags within different languages (*left* Korean, *right* English)

Trelliscope functionality allows us then to compare more languages using multipanel view and see differences, or similarities (we will find of course that many English words or names are used, see Korean example). In general, all of our implemented visualizations on tweets from New York area shows very flexible opportunities in Tessera for a preparation of the interesting application for a data analyst. The main advantages are in aforementioned flexibility and effectiveness of applications in Tessera, possibility to provide many extensions for visualizations into Trelliscope. In the future, we would like to provide more detailed experiments with the scalability of Tessera on real data, and study its possibility to provide a flexible application for visualization of active data streams.

6 Conclusion and Future Work

The main objective of the paper was to introduce the solution for twitter feed analysis and visualization based on R and big data analysis technologies. We used the Tessera framework which was deployed in a small-sized Hadoop cluster environment. Designed visualizations were implemented using Trelliscope and other libraries and available as Shiny web applications. Utilization of the computational capabilities provided by the Hadoop cluster in a wide range of data cleaning, preprocessing and visualization tasks proved to be useful and well-suited for such analytical task performed in R environment. On the other hand, Tessera provides a rich set of functions, scalable on various levels. Trelliscope then proved to be a useful tool for large-scale data visualizations.

Acknowledgments The work presented in this paper was supported by the KEGA project under grant No. 025TUKE-4/2015 and also by the VEGA project under grant No. 1/0493/16.

References

1. White, T.: Hadoop: The Definitive Guide, 1st edn. O'Reilly Media, Inc. (2009)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters, In Sixth Symposium on Operating System Design and Implementation, OSDI'04, pp. 107–113. San Francisco, CA (2004)
3. Tan, Y.S.: Hadoop framework: impact of data organization on performance. J. Softw. Pract. Exp. (2011). ISSN: 0038-0644
4. Vavilapalli, V.K., et. al.: Apache Hadoop YARN: yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13). ACM, New York, Article 5 (2013)
5. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10). Berkeley, CA (2010)
6. Mittal, A., Pathak, S., Bannard, T.: RHadoop: An Improved Execution Environment for Restricted MapReduce Programs (2013)
7. Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B., Cleveland, W.: Large complex data: divide and recombine (D&R) with RHIPE. Stat **1**, 53–67 (2012)
8. Hafen, R., Gosink, L., McDermott, J., Rodland, K., Kleese-Van Dam, K., Cleveland, W.S: Trelliscope: a system for detailed visualization in the deep analysis of large complex. In: Proceedings of the 2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV), pp. 105–112 (2013)

Author Index

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*, Advances in Intelligent Systems and Computing 522, DOI 10.1007/978-3-319-46586-9