Advances in Intelligent Systems and Computing 430

Adam Grzech
Leszek Borzemski
Jerzy Świątek
Zofia Wilimowska *Editors*

Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II

Springer

Advances in Intelligent Systems and Computing

Volume 430

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 · Leszek Borzemski
Jerzy Świątek · Zofia Wilimowska
Editors

Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II

Springer

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

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

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

Zofia Wilimowska
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 978-3-319-28559-7 ISBN 978-3-319-28561-0 (eBook)
DOI 10.1007/978-3-319-28561-0

Library of Congress Control Number: 2016930056

Printed on acid-free paper

This Springer imprint is published by SpringerNature
The registered company is Springer International Publishing AG Switzerland

Preface

This four volume set of books includes the proceedings of the 2015 36th International Conference Information Systems Architecture and Technology (ISAT), or ISAT 2015 for short, held on September 20–22, 2015, in Karpacz, Poland. The conference was organized by the Department of Computer Science and Department of Management Systems, Faculty of Computer Science and Management, Wrocław University of Technology, Poland.

The International Conference Information Systems Architecture is organized by the Wrocław University of Technology from the seventies of the last century. The purpose of the ISAT is to discuss a state of the art of information systems concepts and applications as well as architectures and technologies supporting contemporary information systems. The aim is also to consider an impact of knowledge, information, computing, and communication technologies on managing the organization scope of functionality as well as on enterprise information systems design, implementation, and maintenance processes taking into account various methodological, technological, and technical aspects. It is also devoted to 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 specific disciplinary research, as well as on multi-disciplinary studies 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 people involved in the development of business information systems and business computer applications.

This year, we received 130 papers from 17 countries. The papers included in the four proceeding volumes published by Springer have been subject to a thoroughgoing review process by highly qualified peer reviewers. Each paper was reviewed by at least two members of Program Committee or Board of Reviewers. Only 74 best papers were selected for oral presentation and publication in the

36th International Conference Information Systems Architecture and Technology 2015 proceedings. The final acceptance rate was 57 %.

Professor Peter Nelsen (Denmark) presented his keynote speech on Some Insights from Big Data Research Projects. He also organized the special session on the advances in methods for managing complex planning environments.

The conference proceedings are divided into four volumes and present papers in the areas of managing complex planning environments, systems analysis and modeling, finance, 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.

We would like to thank the Program Committee and external reviewers, essential for reviewing the papers to ensure a high standard of the ISAT 2015 conference and the proceedings. We thank the authors, presenters, and participants of ISAT 2015; without them, the conference could not have taken place. Finally, we thank the organizing team for the efforts this and previous years in bringing the conference to a successful conclusion.

September 2015

Adam Grzech
Leszek Borzemski
Jerzy Świątek
Zofia Wilimowska

ISAT 2015 Conference Organization

General Chair

Leszek Borzemski, Poland

Program Co-chairs

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

Local Organizing Committee

Leszek Borzemski, Chair
Zofia Wilimowska, Vice-Chair
Mariusz Fraś, Conference Secretary and ISAT 2015 Website Administrator
Arkadiusz Górski, Katarzyna Gwóźdź, Technical Editors
Anna Kiłyk, Ziemowit Nowak, Agnieszka Parkitna, Technical Chairmen

International Program Committee

Witold Abramowicz, Poland
Dhiya Al-Jumeily, UK
Iosif Androulidakis, Greece
Patricia Anthony, New Zeland
Zbigniew Banaszak, Poland
Elena N. Benderskaya, Russia
Leszek Borzemski, Poland
Janos Botzheim, Japan

Patrice Boursier, France
Wojciech Cellary, Poland
Haruna Chiroma, Malaysia
Edward Chlebus, Poland
Gloria Cerasela Crisan, Romania
Marilia Curado, Portugal
Czesław Daniłowicz, Poland
Zhaohong Deng, China
Małgorzata Dolińska, Poland
El-Sayed M. El-Alfy, Saudi Arabia
Naoki Fukuta, Japan
Piotr Gawkowski, Poland
Manuel Graña, Spain
Wiesław M. Grudzewski, Poland
Adam Grzech, Poland
Irena Hejduk, 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

Michele Pagano, Italy
George A. 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 2015 Reviewers

Patricia Anthony, New Zeland
Zbigniew Banaszak, Poland
Elena N. Benderskaya, Russia
Grzegorz Bocewicz, Poland
Leszek Borzemski, Poland
Janos Botzheim, Japan
Patrice Boursier, France
Krzysztof Brzostowski, Poland

Wojciech Cellary, Poland
Gloria Cerasela Crisan, Romania
Marilia Curado, Portugal
Mariusz Czekała, Poland
Grzegorz Debita, Poland
El-Sayed M. El-Alfy, Saudi Arabia
Stefan Forlicz, Poland
Mariusz Fraś, Poland
Naoki Fukuta, Japan
Piotr Gawkowski, Poland
Manuel Graña, Spain
Dariusz Gąsior, Poland
Adam Grzech, Poland
Irena Hejduk, Poland
Katsuhiro Honda, Japan
Zbigniew Huzar, Poland
Biju Issac, UK
Jerzy Józefczyk, Poland
Krzysztof Juszczyszyn, Poland
Yannis L. Karnavas, Greece
Radosław Katarzyniak, Poland
Grzegorz Kołaczek, Poland
Zdzisław Kowalczuk, Poland
Binod Kumar, India
Jan Kwiatkowski, Poland
Antonio Latorre, Spain
Arkadiusz Liber, Poland
Wojciech Lorkiewicz, Poland
Zygmunt Mazur, Poland
Pedro Medeiros, Portugal
Izabela Nielsen, Denmark
Peter Nielsen, Denmark
Tadashi Nomoto, Japan
Cezary Orłowski, Poland
Michele Pagano, Italy
George A. Papakostas, Greece
Marek Pawlak, Poland
Jan Platoš, Czech Republic
Tomasz Popławski, Poland
Dolores I. Rexachs, Spain
José S. Reyes, Spain
Gerald Schaefer, UK
Habib Shah, Saudi Arabia
Anna Sikora, Spain
Małgorzata Sterna, Poland

Janusz Stokłosa, Poland
Remo Suppi, Spain
Edward Szczerbicki, Australia
Jerzy Świątek, Poland
Elpida Tzafestas, Greece
José R. Villar, Spain
Tomasz Walkowiak, Poland
Hongzhi Wang, China
Leon S.I. Wang, Taiwan
Adam Wasilewski, Poland
Jan Werewka, Poland
Zofia Wilimowska, Poland
Bernd Wolfinger, Germany
Józef Woźniak, Poland
Jaroslav Zendulka, Czech Republic
Maciej Zięba, Poland
Bernard Ženko, Slovenia

ISAT 2015 Keynote Speaker

Professor Peter Nielsen, Aalborg University, Aalborg, Denmark
Topic: Some Insights from Big Data Research Projects

ISAT 2015 Invited Session

Advances in Methods for Managing Complex Planning Environments
Chair: Peter Nielsen, Denmark

Contents

Part I
Web Systems

Comparison of Relational, Document and Graph Databases in the Context of the Web Application Development

Małgorzata Plechawska-Wójcik and Damian Rykowski

Abstract The paper presents the comparison study of relational, document and graph databases. A social networking web application supporting all three types of database was developed. The analysis consists in data models analysis and performance tests. Data models analysis discusses requirements meeting and difficulties of different implementation. Performance tests present results of obtained for five different tasks. Each of analyzed solutions revealed their strengths and weaknesses. The analysis showed that the graph model the most accurately models the reality. Document database queries occur to be the simplest in use. In terms of performance, PostgreSQL occurred to be the best.

Keywords Relational databases · Document databases · Graph databases · NoSQL

1 Introduction

Technology development and wide access to Internet make it necessary to search for new, more efficient ways to store and process data. Ensuring adequate structure and storage of data is of great importance due to high processing requirements. Over the last decade the amount of data grew rapidly. Their structure often differ from the traditional relational model and new data modeling standards occurred to be necessary. This phenomenon gave rise to the movement called NoSQL (Not only SQL), with main postulates: rejection of the relational model, easy scalability and dynamic structure of data [7]. NoSQL databases suite for distributed systems storing large amount of data. Such conditions reveal limitations of relational databases and exposure potential problems associated with their implementation.

M. Plechawska-Wójcik (✉) · D. Rykowski
Lublin University of Poland, Lublin, Poland
e-mail: m.plechawska@pollub.pl

D. Rykowski
e-mail: rykowski.d@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_1

NoSQL databases gave new opportunities for more efficient data storing, modeling and processing. Their diversity gives also a chance to adjust the optimal solution for a particular problem.

The aim of this paper is to conduct a comparative analysis of three types of databases: relational, documentary and graph in terms of their usefulness in the social network case study. This objective is accomplished based on the example social network web application designed and implemented in order to work with three different databases. Three different type database structures (relational, documentary and graph) were designed and deployed. The implementation covered development of data models and data access layer for three different databases. The paper presents also results of comparative performance tests.

The rest of the paper is structured as follows. Section 2 provides a brief introduction to different database types. Section 3 contains the description of the method applied in the case study. Section 4 presents the example application developed for the purpose of this study. Section 5 provides the details of the case study. It contains the description of the analysis process and presents the results of the analysis. Conclusions are presented in Sect. 6.

2 Relational, Document and Graph Databases

Since the first commercial relational databases were implemented, the relational models completely dominated the way of data storing in applications. However, at the beginning of the twenty-first century, when sudden increase of amount of stored data occurred, relational model with highly structured data often fails especially in the context of performance and scalability [7].

NoSQL is defined as a different than the relational way of storing and processing data. NoSQL suggests a negative attitude to relational database systems and SQL [12, 14]. Performed analyses [9, 13] show that NoSQL databases might supplement or even replace the typical relational solutions in such applications as dealing with medical data [6] or Big Data [3, 8].

2.1 NoSQL Databases

Currently, NoSQL databases are becoming increasingly popular among developers. This interest is not always linked to problems with the system scalability or performance. Most of NoSQL technologies, in contrast to relational databases, have a dynamic data schema [15]. This gives the flexibility to effortlessly add new fields to the database. There is also no need to define a rigid scheme what contributes in easily adapting of new application's requirements. Those features distinguish NoSQL and relational models. Another feature of NoSQL databases is the lack of relationships between data, including foreign keys. Which is associated with the

lack of complex SQL queries and JOIN operations, which are often criticized for their poor performance. Instead, data belonging to one group of objects is selected, and relations between them are handled at the application level [14]. Several types of non-relational databases are available [1, 5]:

- Column-oriented databases store data in columns in contrast to relational databases keeping data in rows.
- Document databases enable to store semi-structured data. Most of document databases uses XML notation: JSON, YAML or BSON.
- Key-value databases enable to store data based on the value of a specified key.
- Graph databases presenting data in the form of connected graph nodes.

Document Databases Document databases store, process and manage data in the form of documents or semi-structured data [8]. The main element of this database type is a document. Depending on the implementation, documents might involve different encapsulation and coding [14]. They provide dynamic data schema, so there is no necessity to predefine it as in the case of relational databases. Each document has a unique identifier to speed up database searching. Because each document can have a different set of fields, classes inherit the base class can extend any number of new fields, which are automatically included in the scheme. In one collection of documents will be kept whole hierarchy of classes [10].

Each document may contain a different set of fields what gives freedom in data modeling. However, it requires more attention from software developers regarding inconsistent and changing data structures. Document databases usually group documents of particular type in collections, what helps to organize an application model.

Document databases support full standardization through the nesting documents and array data type [4]. Because each document might have a different set of fields, it is easy to map inheritance mechanism. Document database does not support JOIN operations, because it could cause declines in performance. The use of nested data instead of dividing the model into several collections has many advantages including avoiding selection of documents from multiple collections and selection performing on the application side as well as assuring atomicity and writing isolation [4]. Despite the many benefits, the use of nested documents has also weakness—such as searching and displaying of nested values is required. To obtain at least one nested object, the entire document needs to be downloaded [2].

Graph Databases Graph databases are data management systems based on a graph structure. They store the data natively in the form of a graph. It provides optimized environment for storing and processing data. Some databases serialize the graph data and write it in a different database type for example relational database. Graph databases apply native graph processing, which physically binds connection nodes [11].

The most common type of graph databases is Property Graph Model [11]. It is characterized by oriented graph composed of nodes and links. Nodes keep properties in the form of key-value pairs. Connections are called, have a direction and

start and end vertex. Another graph model, Hypergraph, may connect any number of nodes. Hypergraphs might be applied to model many-to-many connections.

The most important feature of the graph model are connections, which do not require foreign keys or multiple nested objects. This very simple structure is based on only two type elements enabling to build arbitrarily complex models representing very well the problem domain. These models are usually much simpler and more expressive than NoSQL databases or relational models [11]. Simple and very flexible graph model gives wide abilities to model even complicated data structures [1].

In contrast to relational databases, performance of graph database is relatively constant, independently from amount of stored data. As other non-relational databases, a graph database does not require predefining the database schema. This gives more freedom in data modeling and it often applied in agile software development.

2.2 The Difference Between Relational and Non-relational Databases

The relational databases are based on relational algebra and meet conditions of ACID transactions. NoSQL movement rejects ACID requirements due to problems with the availability and performance of distributed systems. Instead, NoSQL movement supports the set of principles known as BASE [4, 12]:

- Basic availability—any request ensures the answer, the implementation of which may be successful or not.
- Soft state—the system state may vary over time, even in the database does not contain any records.
- Eventual consistency—the system can be temporarily inconsistent, but it will finally gain consistency.

3 Applied Method

Comparison of relational, document and graph databases was performed based on data models analysis and performance tests. To obtain reliable results, a testing application operating on all three database types was developed. The application was used to perform two types of testing:

- Data models analysis. Proper data model design requires knowledge about the problem domain and applied technology. Each discussed in the paper model: relational, document, and graph requires an individual approach. Performed analysis covers discussion about meeting requirements and difficulties of different implementation areas (UserStore, GroupsRepository and PostsRepository).

- Performance tests. Performance tests were conducted for five different operations, where each method was tested on three sets of different size data. The data sets contains a variable number of users and topics, respectively 1000, 5000 and 10,000 records. Each operation was performed 10 times in a loop, only the best time for a given set of data was saved.

4 Testing Application

For testing purposes a web application—a social networking portal was developed. The application uses each of three testing databases respectively: relational, graph and document. The application provides features typical for many social networking portals like friends grouping, private messages or microblogging supplied with the ability to rate and write comments. The portal also enable to create and participate in discussion groups, add tags to topics, rate topics, add comments and perform searching.

The portal was implemented using the ASP.NET MVC 5 framework and C# programming language. The architecture of application was based on the Repository Pattern and the Dependency Injection Pattern. It simplifies the development of three different implementations of repository layer and allows to switch easily between them.

4.1 Implementation of Relational Database Access Layer

PostgreSQL was chosen as a relational database management system. PetaPoco library was applied to manage the data and perform such operations as SQL queries and simple object-relational mapping.

Figure 1 presents entity relationship diagram of the PostgreSQL-based database. Mapping tables are applied to model many-to-many relationships. Mapping tables cause a database scheme to be complicated and difficult to understand.

In the model, all primary keys are of varchar (36) type because dedicated business objects implemented at the application level use 36-chars GUID strings. For better queries performance, indexes were created on all foreign key columns.

4.2 Implementation of Document Database Access Layer

MongoDB was applied in the application as a document database management system. MongoDB team provides official driver for .NET platform and C# language, so all methods implemented in the repository use C# API rather than plain queries

typical for relational database repository. The proper handling of MongoDB driver requires creating of new POCO classes reflecting the structure of documents stored in database. MongoDB provides a special data type dedicated to document identifiers.

Figure 2 presents the document database diagram. Comparing to the relational model it contains less database objects (collections) and it is easier to read. Some relationships is embedded directly in other documents, for example posts and topics contain comments. MongoDB does not support join operations, so data in a database document are denormalized for better performance and easier querying. However, denormalization causes increased number of properties describing documents and might be responsible for problems with data inconsistency.

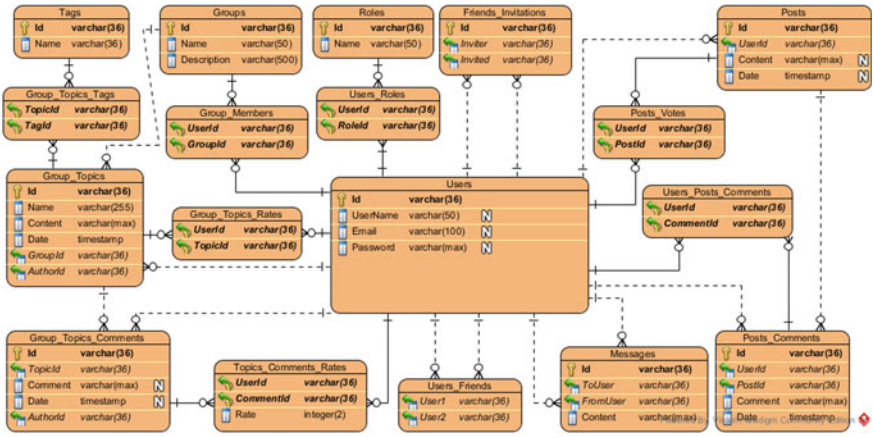

Fig. 1 Diagram of relational database model

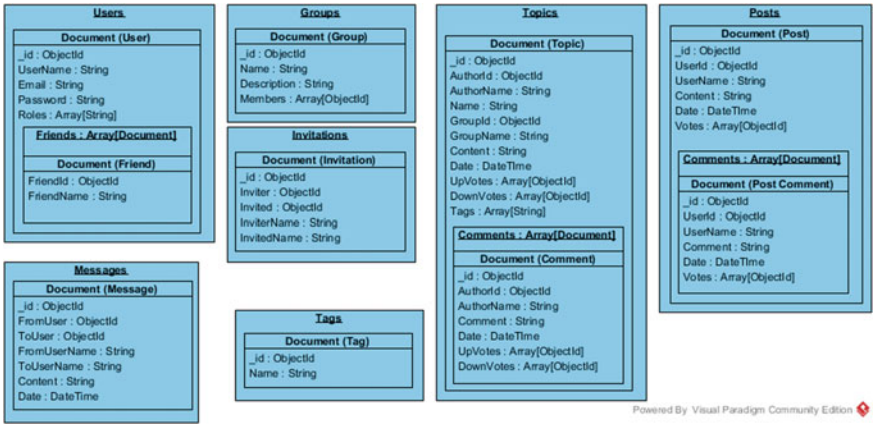

Fig. 2 Diagram of document database model

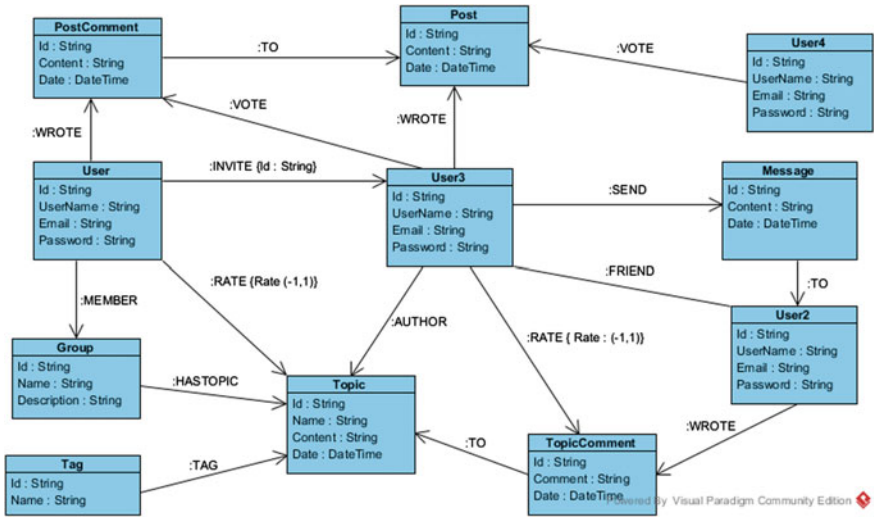

Fig. 3 Diagram of graph database model

4.3 Implementation of Graph Database Access Layer

Neo4j database management system was applied in the task of a graph model implementation. Neo4jClient library was used to run with Neo4j. It provides support functions for constructing Cypher language queries. Neo4jClient allows also to run plain Cypher queries.

Figure 3 presents the graph database diagram. For readability purposes, the diagram contains four User type objects. Graph model looks clear and it is easy to understand. It's advantage is also closeness to reality. Graph model provide for ease of many-to-many relationships modeling without any additional objects. All nodes identifiers are strings to store GUID generated at the application level.

5 Results

5.1 Analysis of Data Models

Analyzing the relational model (Fig. 1) one can see that it contains many more items than the dedicated object model. Flat and tabular structure of a relational database prevents the use of more advanced structures, such as tables or lists. Therefore, it is necessary to introduce intermediate relationships mapping many-to-many relations. In case of large number of relations such model is unreadable and difficult to manage.

Completely different approach is used in the documentary base model. The document structure promotes the object-oriented modeling principles. The idea of nesting documents greatly simplifies the model and reduces the number of model elements. Entities such as Topics and Posts implement natural composition of related objects in a single document (information about comments, evaluation and tags). However, the problem of documentary databases is limited ability to model relationships between documents. All relationships between data need to be handled at the application level. Especially many-to-many relations require development of new collections or nesting in documents a list of related identifiers. The first solution is similar to relational modeling, the second one cause redundancy. Documents with many connections (such as Topics collection) contain a part of the data in the denormalized form in order to avoid combining data from multiple documents on the application side.

The graph data model does not have a problem of excessive number of database objects (as it is in the case of relational databases) or denormalization (as in documentary databases). Graph databases perform well in dealing with highly related data. Native support for many-to-many relations do not require creation of intermediate objects, needed in relational and document models. The graph model seems to be understandable and it better describes the problem domain.

5.2 Performance Tests

Performance tests were performed on five different tasks:

- *Friends of friends*—test was based on the GetPossibleFriends method from repository, which returns possible user friendships.
- *Full-text search*—test checks Full-text search support in tested databases. Searched phrase was word ‘android’.
- *Number of friends*—test was based on the GetFriendsCount method from repository, which returns the number of friends for 100 users.
- *Loading of topic*—test checks performance of GetTopicById method which returns single topic.
- *Rating of topic*—test checks performance of RateTopic method.

Summary results of all performed tests are presented on Table 1.

The most specific test for social networking website is the friends of friends test. Graphical representation of results is shown in Fig. 4.

The best results in friends of friends test have been obtained by the PostgreSQL database. Very good results were also obtained by the Neo4j, which has demonstrated constant execution time of operation. Execution times for the PostgreSQL have demonstrated linear increase. The Full-text search test shows that MongoDB provides the best support for Full-text-searching of the tested databases. Searching in 10,000 topics took only 92 ms. The worst time has been obtained by the PostgreSQL, where searching in the largest data set took 7 s.

Table 1 Summary results of performance tests

Execution time [ms]	PostgreSQL			MongoDB			Neo4j		
Data set	1000	5000	10,000	1000	5000	10,000	1000	5000	10,000
Friends of friends	5	8	11	186	228	242	21	22	22
Full-text search	616	3392	7029	80	80	92	229	866	2454
Number of friends	28	30	30	111	119	119	2840	17,580	32,594
Loading of topic	3	3	4	1	1	1	5	4	46
Rating of topic	3	3	3	6	6	6	11	11	25

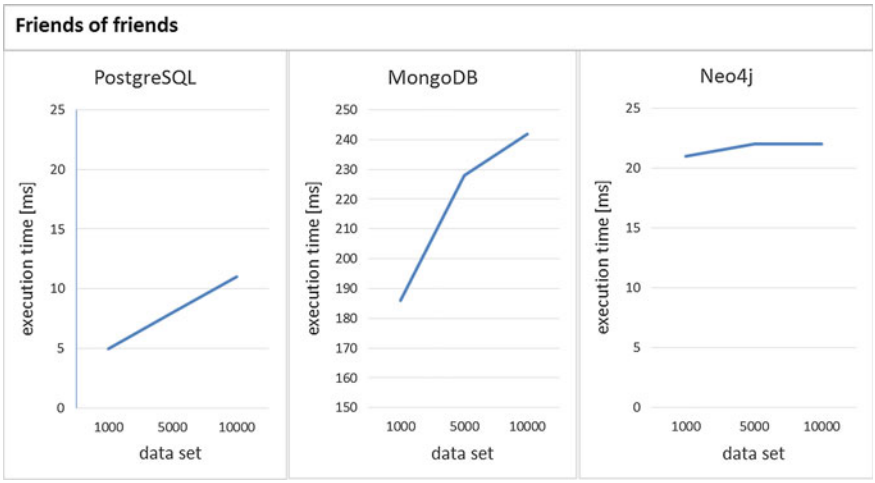

Fig. 4 Friends of friends test results

In the test "Number of friends", PostgreSQL has achieved the best results. Acceptable results have been also obtained by MongoBD. Both databases have had nearly constant execution times for different data sets. Surprisingly bad performance was demonstrated by the Neo4j database. Execution time for the largest data set was 33 s.

In the loading of topic test the best results have been obtained by MongoDB. However all databases have had acceptable results, except Neo4j for 10,000 topics where execution time has increased 10 times.

In the rating of topic test, PostgreSQL again has achieved the best results. Similar to the previous test, all databases have had very good execution times, except Neo4j for the largest data set where execution time has increased 2 times.

6 Summarizing Discussion

The aim of this study was to perform a comparative analysis of three types of databases: relational, documentary and graphs database. The comparison was carried out in terms of their usefulness in social networking application. This objective was achieved based on the design and implementation of a dedicated web application. The comparative analysis did not explicitly indicate which database model occurred the most suitable. Each of analyzed solutions revealed their strengths and weaknesses.

The analysis showed that the graph model the most accurately models the reality. This model has named connections between objects, what makes it more natural and easy to understand. On the other hand the document model, where a single document contains as much useful information as possible, works well in case of dealing with data composition (for example subjects and their comments). However, if the data specificity covers many relationships, the model is highly denormalized. In practice number of relationships between documents should be maximally reduced, because data merging is performed on the application side. The consequence of data denormalization is also the need of data consistency checking. The relational model did not occurred to be adequate in social networking application. In this model many-to-many relationships need to be splitted with intermediate tables. If the model contains many intermediate tables, the schema becomes unreadable and hard to manage. What is mode, the relational database, as the only one among tested solutions, requires a rigid definition of the scheme. The disadvantage of such solution is a long process of the model implementation amd limited dynamic data modeling. However, the big advantage is the validation on the database system level providing high resistance to errors. In graph and document databases the programmer needs to care about data correctness and consistency on the application level.

Other difference lies in query languages: typical SQL in PostgreSQL, C# based query language in MongoDB and Cypher query language for Neo4j. Most of SQL queries applied in the project were simple, but some operations (such as the posts and topics or friends of friends) required complex solutions. MongoDB queries, even complex, were easily implemented and processed. No necessity to consider multiple relationships greatly simplifies the searching process. This database, however, faces problems in simple queries requiring connections of several documents (for example in “friends of friends” task). The Cypher query language for Neo4j database, based on the pattern matching, is very intuitive and easy to use. Cypher, similarly to SQL, is complicated in case of advanced operations including additional data aggregations and manipulation (for example task of tags aggregation for a particular topics).

In terms of performance, PostgreSQL occurred to be the best because it gave the best results in three of the five performed tests. However, the result of Neo4j is constant regardless of the amount of data. In other database systems performing time increases with the amount of data. PostgreSQL, despite the created indexes,

occurred to be weak in the full text searching task. MongoDB gave poor results in tasks needing connection of several documents at the application level. Neo4j database surprisingly failed to optimally complete the task requiring large number nodes aggregation.

The social network service has very specific data model regarding simultaneously many graph-shaped connections to and more tabular structures. None of compared models did not fully meet the requirements. The ideal solution would implement combination of multiple models. The obtained results show that finding relevant solutions adequate to a particular model of reality is still a big challenge.

References

1. Buerli, M.: The Current State of Graph Databases. San Luis Obispo (2012)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Google, Inc. http://research.google.com/archive/mapreduce-osdi04-slides/index.html
3. Dziedzic, A., Mulawka, J.: Analysis and comparison of NoSQL databases with an introduction to consistent references in Big Data storage systems. In: Symposium on Photonics Applications in Astronomy, Communications, Industry and High-Energy Physics Experiments, p. 92902V. International Society for Optics and Photonics (2014)
4. Copeland, R.: MongoDB Applied Design Patterns. O'Reilly Media, Inc., Sebastopol (2013)
5. Ercan, M., Lane, M.: Evaluation of NoSQL databases for EHR systems. In: 25th Australasian Conference on Information Systems, Auckland, Nowa Zelandia (2014)
6. Lee, K., Tang, W., Choi, K.: Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage. Comput. Methods Programs Biomed. **110**(1), 99–109 (2013)
7. Mohan, C.: History Repeats Itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla. EDBT/ICDT '13, Genoa, Włochy (2013)
8. Moniruzzaman, A., Hossain, S.: NoSQL database: new era of databases for big data analytics —Classification, characteristics and comparison. Int. J. Database Theory Appl. **6**(4) (2013)
9. Nimis, J., Kammerer, M., Armbruster, M., Mattes, C., Steinbuch, K.: Application-mimes an approach for quantitative comparison of SQL- and NoSQL-databases. In: Proceedings of the 16th International Conference on Enterprise Information Systems, vol. 1, pp. 256–263 (2014)
10. Plugge, E., Membrey, P., Hawkins, T.: Definitive Guide to MongoDB. Apress, New York (2010)
11. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media, Inc., Sebastopol (2013)
12. Sharma, V.: Meenu Dave. SQL and NoSQL databases. Int. J. Adv. Res. Comput. Sci. Software Eng. **2**(8) (2012)
13. Upeksha, D., Wijayarathna, C., Siriwardena, M., Lasandun, L., Wimalasuriya, C., de Silva, N., Dias, G.: Comparison between performance of various database systems for implementing a language corpus. In: Beyond Databases, Architectures and Structures, pp. 82–91. Springer, Berlin (2015)
14. Vaish, G.: Getting Started with NoSQL. Packt Publishing, Birmingham (2013)
15. Knowledge Base of Relational and NoSQL Database Management Systems. http://db-engines.com/

A Cluster-Based Quality Aware Web System

Krzysztof Zatwarnicki, Maciej Płatek and Anna Zatwarnicka

Abstract Providing high quality of Web services is now very important for most of the services. In this article, we present the MLF (Most Loaded First) system, adaptive and intelligent cluster-based Web system. The MLF system provides Quality of Web Service (QoWS) on a fixed level, guaranteeing the page response time within established boundaries, even if Web servers are heavy loaded. We present experiments conducted on real cluster of Web servers and show that MLF system is more reliable than other examined systems.

Keywords Quality of web services · Guaranteeing web page response time · HTTP request scheduling · Request distribution

1 Introduction

The Internet as the medium of information gained popularity at the beginning of the 21st century. The increase of amount of Internet users brought problems with service of large number of clients to many Web services. This resulted in rapid development of Web system technology.

In the past few years most of the works was focused on the Web systems which are very efficient, and able to handle numerous of incoming HTTP requests. Nowadays Internet systems are not only used for entertainment and information purpose but also in business and government administration. This causes the Web

K. Zatwarnicki (✉) · M. Płatek · A. Zatwarnicka
Department of Electroengineering, Automatic Control and Computer Science, Opole University of Technology, ul. Prószkowska 76, 45-758 Opole, Poland
e-mail: k.zatwarnicki@gmail.com

M. Płatek
e-mail: mac.platek@gmail.com

A. Zatwarnicka
e-mail: anna.zatwarnicka@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_2

systems should not only be efficient but also reliable and able to service the user in demanded time [6].

The most common way to build highly efficient Internet system is to use cluster of servers, which give the opportunity to scale the system to the needs. The problem of constricting highly efficient cluster-based Web systems was already discussed in our previous works. We have designed a locally distributed Web cluster system [14], a globally distributed Web cluster system with a broker [4] and a globally distributed Web cluster system without a broker [13].

In our later work we proposed new construction of cluster-based MLF Web system providing fixed QoWS [16]. The system was evaluated in simulation experiments which have shown that it works better than other well-known cluster-based systems. This article presents the results of experiments conducted for the MLF system with the use of real Web servers. We analyze these results to determine if the system can provide quality of the service on demanded level.

The proposed system keeps the page response time within established boundaries in such a way that with a heavy workload, the page response times, for both small and complex pages, would not exceed the imposed time limit.

There have been many works on how to guarantee Web service quality. Most of them concern maintaining the quality of the service for individual HTTP requests [1, 2, 5, 7, 8, 10]. Very few papers were dedicated to the problem of designing Web service, which would guarantee servicing the whole WWW pages within a limited time.

QoWS often is combined with granting privileges to certain group of users [11, 12]. The MLF system proposed in this article not only provides fixed QoWS, keeping the page response time within established boundaries, but also treats all users equally.

The paper is divided into four sections. Section 2 describes the MLF system and the method of scheduling and distributing HTTP requests. Section 3 presents the conducted experiments with results, and Sect. 4 contains concluding remarks.

2 MLF System

The MLF system consists of (Fig. 1a): clients sending HTTP requests, MLF Web switch and Web servers. Web Switch and Web servers with database form a Web cluster. The Web switch task is to receive all of the requests from clients, queue them and send into chosen Web servers. The Web server services requests with accompany of database servers and send requests back to the clients.

The Web switch is the most crucial element of the system because it controls the operations of the cluster. Its main task is to deliver entire Web page with the fixed time tmax. To achieve this, the Web switch calculates the term in which each of the incoming requests have to be serviced on the Web server. The requests are queued in the switch according to these calculated terms. When each request leaves the

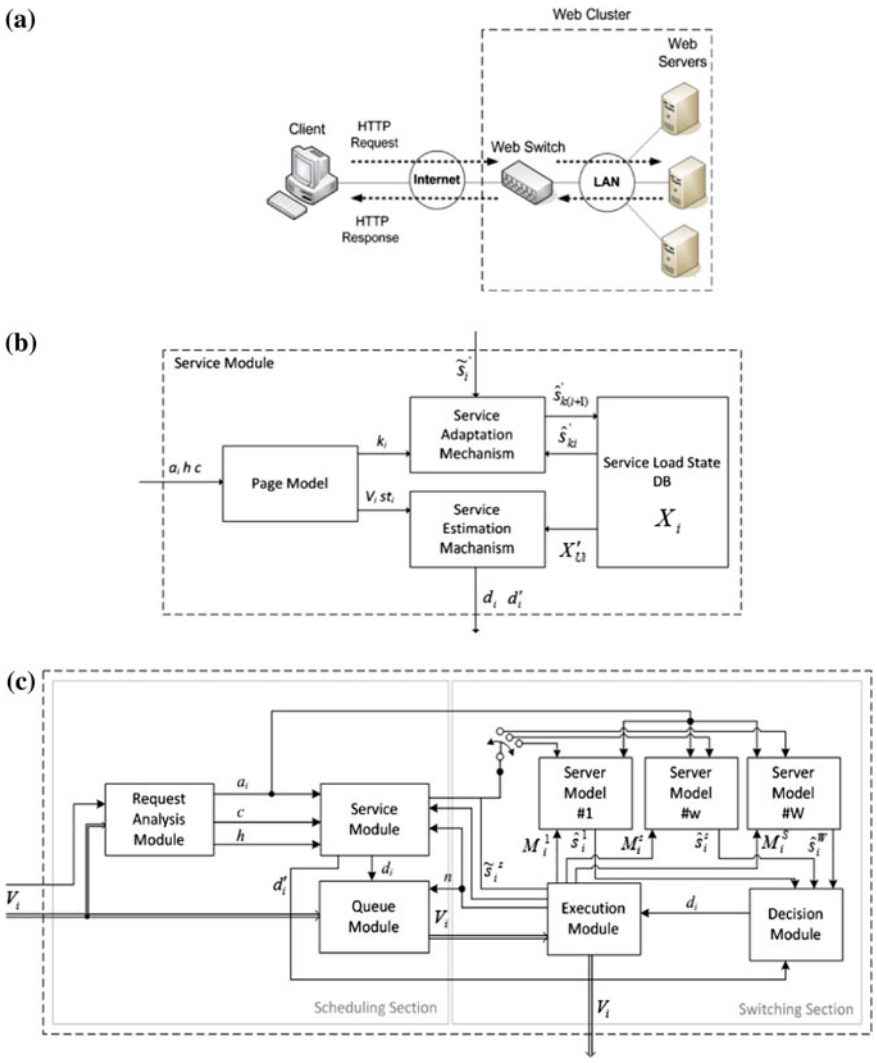

Fig. 1 MLF system: **a** overall view, **b** web switch, **c** service module

queue, the Web switch determines which Web server is able to service the request in required time.

The MLF Web switch consist of two logical sections (Fig. 1c). First, scheduling section is responsible for scheduling request. The second one is a switch section and its role is to distribute requests among Web servers.

Upon receiving incoming HTTP request ri, where i is the request index, the Web switch passes it to the Request Analysis Module (RAM). At the RAM following information are fetched: address ai of requested object, client identifier c and page

identifier h where the requested object belongs. Both identifier h and c are fetched from cookie section of the HTTP request. After receiving all information from RAM the Service Module (SM) computes deadline di which specify when service of the HTTP request has to be started, and d'_i which is the moment when the request service has to come to the end (Fig. 1b). When deadline di is successful calculated, it is used in Queue Module (QM) to queue request according to Earliest Deadline First policy (request are prioritized based on deadline di).

Request skips QM if number of currently serviced requests is smaller than n_{max}, which stand for the lowest value of requests serviced by web servers in the cluster, for which the service reaches the maximal throughput. Primary element of scheduling section is the SM, which is composed of Page Model (PM), Service Load State Database (SLSDB), Service Estimation Mechanism (SEM) and Service Adaptation Mechanism (SAM). The PM stores information about structure of Web pages serviced by the Web cluster, the HTTP clients and the classes of that objects. Basing on a_i, h and c the PM specifies time st_i and vector V_i. Time st_i is measured from the moment the client requests the first object which belongs to the page h to the moment the request u_i arrives. Vector $V_i = \left[k_i^1, \ldots, k_i^l, \ldots, k_i^L\right]$ stores information about classes of objects of page not jet downloaded by the cth client. Classes are composed based on basic information about object. Static objects (files) are partitioned into classes on the base of its size and each of dynamic object (created at the request arrival) has its own class. Object belonging to the same class have similar service time.

The SLSDB stores information about service times for different classes $Z'_i = [\hat{s}'_{1i}, \ldots, \hat{s}'_{ki}, \ldots, \hat{s}'_{Ki}]$, where $\hat{s}'_{ki}$ is the estimated time to service request from specified kth class.

SEM uses information $Z'_{Vi} = [\hat{s}'_{1i}, \ldots, \hat{s}'_{ki}, \ldots, \hat{s}'_{Ki}]$ about service times of objects pointed in the V_i to compute d_i and d'_i. The deadline d_i is calculated as follows: $d_i = \tau_i^{(1)} + \Delta d_i - \hat{s}_{1i}$, where $\tau_i^{(1)}$ is the time of arrival of the ith request, and $\Delta d_i = \hat{s}'_{k^1 i}(t_{\max} - h_i) \Big/ \left(\lambda \sum_{l=1}^{L} \hat{s}'_{k^l i}\right)$ is the period of time which the request can spend being queued and serviced by the Web server. The λ is a concurrency factor, and it depends on the number of Web objects currently downloaded for given Web page. According to [15], the value of this factor for average Web pages can be set to $\lambda = 0.267$. The term d'_i is calculated in the following way $d'_i = \tau_i^{(1)} + \Delta d_i$, and it is the term the service of request has to be done by the Web server.

The SAM module is responsible for adapting the information X'_i. Service time for each class of request is updated after queued request leaves queue for service. The modification is calculated as follows: $\hat{s}'_{k\,(i+1)} = \hat{s}'_{k\,i} + \hat{\eta}(\tilde{s}_i - \hat{s}'_{k\,i})$, where $\tilde{s}_i$ is a measured value of the service time, and $\hat{\eta}$ is an adaptation factor.

When the request r_i leaves the QM it is directed to the switching section, which consist server model modules (SMM), a decision module (DM), and an execution module (EM).

SMM's are assigned one-to-one into Web servers working in the cluster. The SMM is responsible for estimation of the request service time $\hat{s}_i^w$ for the Web server the module is assigned to, where $w \in \{1, \ldots, W\}$ is the server index, and W is the number of Web servers in the cluster. The service time is computed on the base of information of the requested object class k_i and the load $M_i^w = \left[e_i^w, f_i^w\right]$ of the server the SMM is assigned to, where e_i^w is the number of requests being concurrently serviced by the wth server and f_i^w is the number of dynamic request concurrently serviced.

The DM chooses the Web server z_i to service the request. The decision is made in the following way:

$$z_i = \begin{cases} \min\left\{w\text{: } w \in \{1, \ldots, W\} \wedge \Delta d_i' \leq \hat{s}_i^w\right\} \text{ if } r = r_{\max} \text{ and } \underset{w \in \{1,\ldots,W\}}{\exists} \hat{s}_i^w \geq \Delta d_i' \\ w_{\min}\text{: } \hat{s}_i^{w_{\min}} = \min\left\{\hat{s}_i^w : \ w \in \{1, \ldots, W\}\right\} \text{ in other case} \end{cases}, \tag{1}$$

where $\Delta d_i' = \tau_i^{(2)} - d_i'$, and $\tau_i^{(2)}$ is the moment the request r_i leaves the QM. Using formula (1) the DM chooses the server with the lowest index, which is able to service the request in a time not longer than $\Delta d'$. If such server is not available, it offers the server with the shortest service time. Proposed solution guarantee that the Web servers with the lowest indexes are the most loaded, but still have ability to service requests, and the Web servers with the higher indexes are unloaded and able to handle requests in a short time.

After the decision z_i is taken, the EM sends the request r_i to the chosen server. The module also collects information e_i^w and f_i^w, and measures service time $\tilde{s}_i$.

The SMM is the most complex module of the switching section. It use a neuro-fuzzy model to estimate service time $\hat{s}_i^w$ of the request for the given Web server (Fig. 2a). For each class $k_i = 1, \ldots, K$ of objects the same model is used, but with different parameters (weights). Parameter database $Z_i = [Z_{1i}, \ldots, Z_{ki}, \ldots, Z_{Ki}]$ stores parameters for different classes, where $Z_{ki} = [C_{ki}, D_{ki}, S_{ki}]$, $C_{ki} = \left[c_{1\,ki}, \ldots, c_{l\,ki}, \ldots, c_{(L-1)\,ki}\right]$ and $D_{ki} = \left[d_{1\,ki}, \ldots, d_{m\,ki}, \ldots, d_{(M-1)\,ki}\right]$ are parameters of input fuzzy set functions, and $S_{ki} = \left[s_{1\,ki}, \ldots, s_{j\,ki}, \ldots, s_{J\,ki}\right]$ are parameters of output fuzzy set functions. Input fuzzy set functions are triangular (Fig. 2b) and are denoted as $\mu_{F_{el}}(e_i)$, $\mu_{F_{fm}}(f_i)$, $l = 1, \ldots, L$, $m = 1, \ldots, M$, whereas output fuzzy sets functions $\mu_{Sj}(s)$ are singletons (Fig. 2c). The values L and M were chosen experimentally and set to 5, and $J = L \cdot M$.

The service time is calculated in the following way: $\hat{s}_i = \sum_{j=1}^{J} s_{jki} \mu_{R_j}(e_i, f_i)$, where $\mu_{R_j}(e_i, f_i) = \mu_{F_{el}}(e_i) \cdot \mu_{F_{fm}}(f_i)$. The values of parameters C_{ki}, D_{ki}, S_{ki} are modified in an adaptation process using the Back Propagation Method each time the service of the request on a given server finishes. The parameters of output fuzzy sets are modified as follow: $s_{j\,k(i+1)} = s_{j\,ki} + \eta_s \cdot (\tilde{s}_i - \hat{s}_i) \cdot \mu_{R_j}(e_i, f_i)$, whereas parameters of input fuzzy sets are computed in following way $c_{\varphi k(i+1)} = c_{\varphi ki} + \eta_c(\tilde{s}_i - \hat{s}_i)$

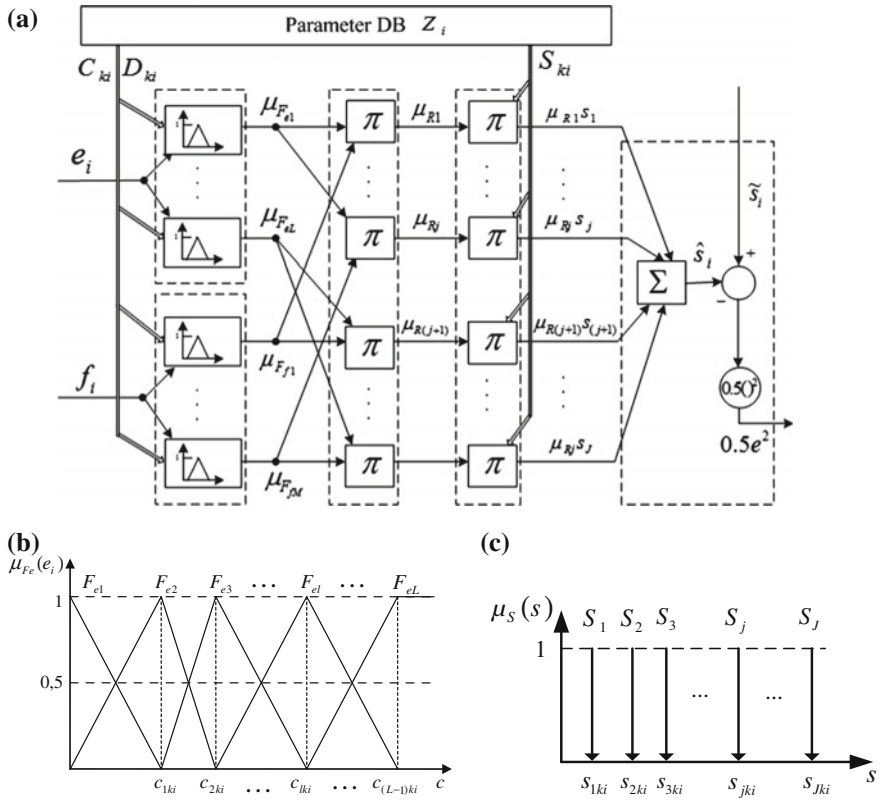

Fig. 2 Server model: **a** neuro-fuzzy model, **b** input fuzzy sets functions, **c** output fuzzy sets functions

$\sum_{m=1}^{M}\left(\mu_{F_{fm}}(f_i)\sum_{l=1}^{L}\left(s_{((m-1)\cdot L+l)ki}\partial\mu_{F_{el}}(e_i)/\partial c_{\varphi ki}\right)\right)$ and $d_{\gamma k(i+1)} = d_{\gamma ki} + \eta_d$ $(\tilde{s}_i - \hat{s}_i)\sum_{l=1}^{L}\left(\mu_{F_{el}}(e_i)\sum_{m=1}^{M}\left(s_{((l-1)\cdot M+m)ki}\partial\mu_{F_{fm}}(f_i)\Big/\partial d_{\gamma ki}\right)\right)$, where η_s, η_c,η_d are adaptation ratios, $\varphi = 1,\ldots,L-1$, $\gamma = 1,\ldots,M-1$.

3 Testbed and Results of Experiments

Our previous research [16] shows that MLF system can provide higher QoWS than other reference distribution methods. The main goal of presented in this article experiments was to evaluate the MLF system using real Web server and the Web switch. We tried to determine if MLF system can provide service with demanded quality of the service.

The experiments were conducted with use of six computers. First of them was responsible for simulating the behavior of Web clients. The next four were Web servers and the last one was a Web switch. Table 1 presents their hardware configuration.

Web servers had the lowest computational power which allowed to reach maximal number of requests they can handle without overloading other elements of the system. All computers were connected to LAN network using gigabyte Repotec RP-G3224V network switch.

The Web server hosted five different Web pages. Table 2 presents the structure of the pages.

Pages contained from 10 to 30 embedded objects with size from 1 to 100 KB and the dynamic pages used PHP as the script language to generate the content of the page. Moreover one page used SQL requests to the MySQL database which contain three related tables containing 10,000 rows each.

Software used in Web switch was written in the C++ language with the use of the following libraries: *libsoup* [9] to implement virtual Web server managing incoming HTTP requests and *boost* [3] for time measuring.

Along with the MLF method other popular distribution methods were implemented in the Web switch:

Table 1 Hardware configuration of computers used in experiments

Name	CPU	Operating system
Web servers	Intel Celeron 1.7 GHz	Ubuntu 13.04 Server
Web switch	Intel Core i7-2670 2.2 GHz	Fedora 18
Client simulator	Intel Core i5-3470 3.3 GHz	Ubuntu 13.04 Desktop

Table 2 Web pages used in the experiments

Name	Type and size of the frame object of the page	Embedded objects number and sizes	MySQL database
Static 10	Static 1 KB	10, size 1–100 KB, sum 477 KB	–
Static 30	Static 1 KB	30, size 1–100 KB, sum 1.39 MB	–
Dynamic 10	Dynamic, PHP	10, size 1–100 KB, sum 477 MB	–
Dynamic 30	Dynamic, PHP	30, size 1–100 KB, sum 1.39 MB	–
Dynamic MySQL 30	Dynamic, PHP	30, size 1–100 KB, sum 1.39 MB	Requests to database containing 3 tables, 10,000 rows each

- LARD (Locality-Aware Request Distribution): one of the best distribution methods taking into account localization of the previously requested object,
- CAP (Content Aware Policy): an algorithm uniformly distributing HTTP requests of different types,
- RR (Round-Robin): an algorithm distributing uniformly all incoming requests.

Simulation of clients was done by HTTP Request Generator (RG), which was written in the C++ language with the use of the *LibcURL* library. The way of working of the RG was similar to way of operation of modern Web browsers. Each client generated by RG was opening up to 6 TCP connections to download single Web page.

Except of generating HTTP requests, the RG was able to collect information about requests, such as: mean value of request response time and satisfaction. Satisfaction is the measure of user's satisfaction of the page response time. The user's satisfaction depends on the response time of the whole page according to function presented on Fig. 3. Value of the satisfaction is equal to 1 when the page response time is shorter than $t^s_{\max}$, and reaches value 0 when the time is longer than $t^h_{\max}$. In all of the experiments following values of $t^s_{\max} = t_{\max} = 2000$ ms and $t^h_{\max} = 2t^s_{\max}$ were adopted.

The Fig. 4a presents diagram of the mean request response time vs. the load for different Web pages. Diagrams from b to f in Fig. 4 present satisfaction vs. number of clients generating HTTP requests. As one can see the highest value of satisfaction was obtained by the MLF method for static small pages as well as pages containing more embedded objects. Results for dynamic pages are not spectacular however in every experiment the satisfaction is higher for the MLF then for other methods. Results presenting the mean request response time shows that for the MLF method not only the satisfaction is the highest but also the response time is no longer than for other methods.

The presented results confirm results obtained during simulations and it can be concluded that using MLF method can help to provide QoWS on a fixed level in the cluster-based Web systems.

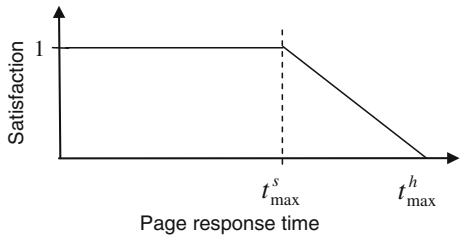

Fig. 3 Satisfaction function

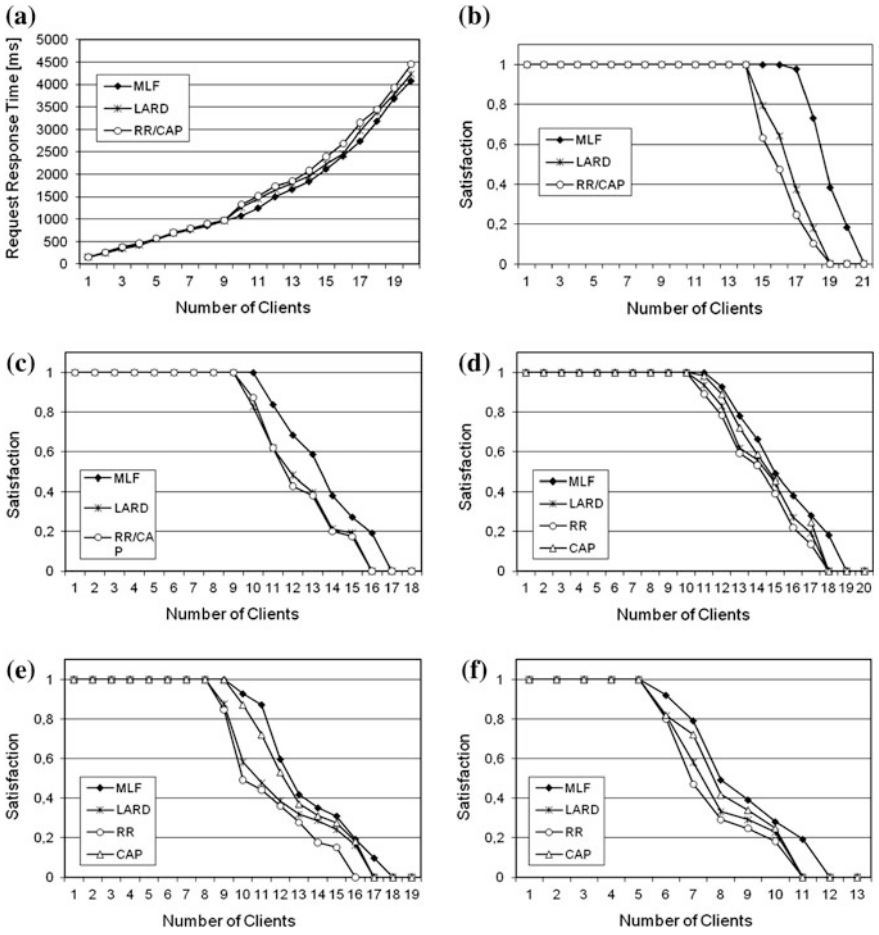

Fig. 4 The results of experiments. **a** Mean request response time for Static 10; Satisfaction versus load for the Web pages: **b** static 10, **c** static 30, **d** dynamic 10, **e** dynamic 30, **f** dynamic MySQL 30

4 Summary

In this paper we presented HTTP request scheduling and distribution cluster-based Web system, which provide service at fixed level. The proposed MLF system can deliver requested content for the users with higher value of satisfaction than other examined methods. Decision algorithms, used in the MLF method, applies adaptive algorithms using neuro-fuzzy models in its construction. Obtained results confirms experiments conducted during simulations and presented in other articles. At high load on Web servers, the MLF system can provide acceptable value of satisfaction for higher number of clients than other tested methods. When comparing the same load for each method it is noticeable that MLF system supplies HTTP requests in

lower response times, causing higher values of satisfaction in each case. The results of experiments show that MLF method can makes Web system more reliable than it will be under control of other examined methods.

References

1. Abdelzaher, T.F., Shin, K.G., Bhatti, N.: Performance guarantees for web server end-systems: a control-theoretical approach. IEEE Trans. Parallel Distrib Syst **13**(1), 80–96 (2002)
2. Blanquer, J.M., Batchelli, A., Schauser K., Wolski R.: Quorum: Flexible quality of service for internet services. In: Proceedings of Symposium on Networked Systems Design and Implementation (2005)
3. Boost C++ libraries, http://www.boost.org/. Access 10 Sept 2015
4. Borzemski, L., Zatwarnicka, A., Zatwarnicki, K.: Global Adaptive Request Distribution with broker, Vol. 4693, pp. 271–278. Lecture Notes in Artificial Intelligence. Springer, Berlin (2007)
5. Harchol-Balter, M., Schroeder, B., Bansal, N., Agrawal, M.: Size-based scheduling to improve web performance. ACM Trans. Comput. Syst. **21**(2), 207–233 (2003)
6. Hunek, W.P., Dzierwa, P.: New results in generalized minimum variance control of computer networks. J. Inf. Technol. Control **43**(3), 315–320 (2014)
7. Kamra, A., Misra, V., Nahum, E.: Yaksha: A self tubing controller for managing the performance of 3-tiered websites. In: Proceedings of International Workshop on Quality of Service, pp. 47–56 (2004)
8. Kanodia, V., Knightly, E.: Multi-class latency-bounded web services. In Poceedings of te 8th International Workshop on Quality of Service (IWQOS) (2000)
9. Libsoup library description, http://developer.gnome.org/libsoup/stable/. Access 10 Sept 2015
10. Schroeder, B., Harchol-Balter, M.: Web servers under overload: how scheduling can help. In: 18th International Teletraffic Congress, Berlin, Germany (2003)
11. Suchacka, G., Borzemski, L.: Simulation-based performancestudy of e-commerce Web server system-results for FIFO scheduling. In: Multimedia and Internet Systems: Theory and Practice, AISC, Vol. 183, pp. 249–259. Springer, Berlin (2013)
12. Wie, J., Xue, C.Z., QoS: Provisioning of client-perceived end-to-end QoS guarantees in web servers. IEEE Trans. Comput. **55**(12) (2006)
13. Zatwarnicki, K.: Neuro-Fuzzy Models in Global HTTP Request Distribution, Vol. 6421/2010, pp. 1–10. Lecture Notes in Computer Science. Springer, Berlin (2010)
14. Zatwarnicki, K.: Adaptive control of cluster-based web systems using neuro-fuzzy models. Int. J. Appl. Math. Comput. Sci. (AMCS) **22**(2), 365–377 (2012)
15. Zatwarnicki, K., Zatwarnicka, A.: Estimation of Web Page Download Time, Communication in Computer and Information Science, Vol. 291/2012, pp. 144–152. Springer, Berlin (2012)
16. Zatwarnicki, K.: Operation of Cluster-Based Web System Guaranteeing Web Page Response Time, Vol. 8083/2013, pp. 477–486. Lecture Notes in Artificial Intelligence. Springer, Berlin (2013)

Comparison of Text-Similarity Metrics for the Purpose of Identifying Identical Web Pages During Automated Web Application Testing

Marek Zachara and Dariusz Pałka

Abstract The paper focuses on the evaluation of effectiveness of a number of algorithms used to assess text similarity. The purpose of such evaluation is to determine the best methods for comparing and identifying near-identical web pages. Such comparison of web pages is in turn a prerequisite for building new automated testing tools and security scanners. The goal is to build scanners that will be able to automatically test the web application behavior for a large range of supplied parameters (known as *fuzzing*). Such testing requires massive generation and processing of requests, which in turn require fast page comparison methods. The similarity comparison is performed on a shortened, tokenized version of web pages, using a test set of pages downloaded from popular websites. A methodology for the evaluation of similarity metrics is proposed, together with a quality metric for the intended task. Several tokenization strategies are also tested and their impact on the final result is assessed.

Keywords Web pages similarity · Similarity metrics · Automated website testing

1 Introduction

This paper focuses on identifying the best method for fast comparisons of two arbitrary web pages and assessing their similarity. The comparisons are done on transformed, tokenized web pages, utilizing string similarity metrics to assess the pages similarity. A number of known string metrics are compared in order to evaluate their fitness for the purpose.

M. Zachara (✉) · D. Pałka
AGH University of Science and Technology, 30 Mickiewicza Avenue, Krakow, Poland
e-mail: mzachara@agh.edu.pl

D. Pałka
e-mail: dpalka@agh.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_3

1.1 Rationale

The ability to compare two web pages and judge their similarity has a number of obvious applications. Identifying plagiarism [11] and phishing web sites (web sites build to be visually identical to the targeted legitimate site of e.g. a bank) [14, 20] is an obvious choice.

Another use for the classification of web pages with identical or very similar content is the elimination of duplicate data during the process of web data mining, including elimination of duplicate web search results [6].

There is however, another application, which constitutes the background for the research presented in this article, namely reliable traversing and indexing of dynamically-generated web sites.

Dynamically generated web pages/websites are the ones that construct and provide response to an HTTP request in real time, based on a number of factors:

- the parameters sent by the client to the server as a part of GET or POST request
- the state of the user's session (if any)
- the internal state of the application
- data provided from outside sources (e.g. advertisements)

The resultant HTML code, sent to the client, is usually generated with a specialized server-side framework and a programming language, of which Java (J2SE, J2EE, JSP etc.), PHP and ASP are probably the most common. Because a web page is generated for each request separately, the actual content of the same web page may differ (e.g. it can have a different advertisement or new users' comments etc.). As a result, it may happen that different web pages will be sent to a user for the same URL, but also identical web pages may be available at different URLs.

A typical method of indexing (i.e. identifying all distinct web pages) of a website, also known as *crawling*, relies on retrieving all unique and not yet visited URLs from the HTML of the analyzed web pages, and visiting them in turn until the list is exhausted. For dynamically generated web pages, this may often not work or work poorly because of the issues described above. In some extreme cases, the whole web application may be accessible via only a single URL.

While indexing web sites by search engines is usually in the interest of the web-site owner, and he or she has an incentive to build an application in such a way that it is correctly indexed, there is another area of use for the web-site crawling and identifying of all the web pages—and this is automated compliance and security testing.

Due to the rapid growth in number and complexity of web applications, it is virtually impossible to manually test them thoroughly after each change or upgrade. At the same time, the number of vulnerabilities in them is soaring. To provide just one example: in its recent report, Symantec estimates the number of vulnerable applications to be at 78 %, with 16 % having critical vulnerabilities [19]. Development of automated methods for traversing the target website and identifying all unusual behavior is therefore an important challenge.

A part of this effort lies in identifying reliable and fast tools for comparison of web pages, as this will allow for building automated scanners for a number of typical vulnerabilities, including path traversal, manipulation of parameters data, etc.

1.2 Related Work

When comparing two web pages and assessing their similarities, three basic approaches are usually employed.

One is to compare the pages' structure, usually in the form of the resultant DOM model against each other. This approach is discussed in [1, 15, 20]. The typical approach is to use tree or graph comparison methods– to identify identical sub-graphs (or sub-trees). This approach is however computationally intensive, as all graph isomorphism problems are.

The other is to visually compare the layout of the pages, using various image processing tolls, like the ones described in [4]. This, however, is also a very computationally intensive process. Both of these approaches are currently not feasible for mass-scale comparisons.

The third approach relies on comparing the HTML code of the web pages, utilizing text-related tools and methods, mainly string similarity metrics.

Starting from a simple use of Levenshtein distance in [10], a number of methods were tried utilizing specific similarity metrics [6] or bi- and tri-grams of words [14]. There are a few published comparisons of the sting metrics with their effectiveness evaluated for the specific purpose, like names matching [3] or short advertisement matching [22]. The specific context of use required, however, a dedicated investigation targeted at the defined task.

Text (HTML) based comparisons are much faster as they do not require the resource-intensive task of rendering web pages, but they may not recognize similarly looking web pages if they have a different html structure. This is not an issue for automated testing of a web-service, as the layout of the pages tested is (in most cases) expected to adhere to a common pattern. It is therefore advisable to utilize these methods for the intended task. There are quite a few various text-similarity metrics developed over time, but there seem not to be any comprehensive comparison of their effectiveness for the task of assessing the similarity of web pages. This paper focuses on this task, comparing the results of various similarity metrics on a large set of web pages.

2 Methods and the Scope of Evaluation

With the constant growth of the infrastructure capacity and the functionality of the web services, the size of web pages grew from a few to hundreds of kilobytes (see Table 1). Comparing the pages by character is therefore ineffective, since even the

most efficient algorithms recommended in [12] have computational complexity of $n * m/\log(n)$.

A quick practical experiment showed that calculating Levenshtein distance for such pages can take a couple of minutes (for single-threaded implementation). In the web page comparison, the similarity is often assessed based on a tokenized representation of the data. Such approach is discussed in [10], where the similarity is computed on the basis of Levenshtein [9] distance between abbreviated tags and tag parameters. Another paper [6] evaluates the performance of a number of algorithms also based on tokenized comparisons.

The same approach is used in the presented research, although the tokenization process is more general and applies to all elements of the assessed page; i.e. both the tag structure and the text content. The process is illustrated in Fig. 1.

The resultant tokens were 1–4 character in length and the impact of the length of the token on the quality of the comparison was investigated. As the result of this technique, the size of the compared text was reduced approx. 20–50 times (1000–3000 tokens for 100,000 characters of the HTML, see Table 1).

The primary goal of the research was to evaluate a large number of text comparison algorithms. For this purpose the Simmetrics library was used [16], as well as Richard Clayton's similarity metrics library [2].

Table 1 Sample websites' metrics

Website	Description	Html size	Tags	Text items	Text length
reddit.com	Home page	110,368	1663	1286	10,295
edition.cnn.com	Home page	202,598	1648	1043	6481
edition.cnn.com	Science	74,426	701	434	5562
bbc.co.uk	Home page	101,884	466	352	2251
bbc.com	Travel	125,847	853	549	8383
businessinsider.com	Home page	143,913	1388	950	7001
businessinsider.com	Lifestyle	122,312	1106	663	7305
amazon.com	Home page	423,356	2020	1046	10,553
amazon.com	A book offer	414,857	2932	2196	29,995

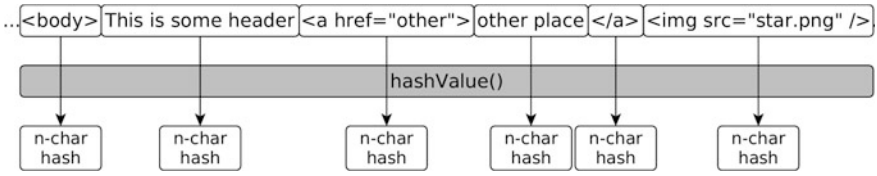

Fig. 1 Generating tokens from the HTML data

2.1 Comparison Methodology

For the purpose of the research, a corpus of several hundred pages was prepared, downloaded from popular websites: cnn.com, bbc.co.uk, gawker.com, businessinsider.com, huffingtonpost.com, amazon.com. For each of these sites, a home page was retrieved as well as a number of sub-pages. Each page was retrieved repeatedly over a period of approximately one month to provide almost identical, yet slightly different pages.

A good similarity assessment algorithm should be able to both correctly identify near-identical web pages and differentiate distinct pages. Therefore, a few different scenarios were evaluated:

1. **Same page comparison**. To identify algorithms' ability to identify near-identical pages, each page of the test corpus was retrieved multiple times from the target website over a course of time. Each of these copies differs slightly, because web pages change over time—due to different advertisements displayed, user comments posted on site and other reasons.
2. **Sibling comparison**. In this scenario, the algorithms were tried with two different sub-pages, both from the same website. Since the sub-pages at the same website usually have the same layout, this is the most difficult test for the algorithms to pass.
3. **Home page comparison**. In this scenario, the algorithms were supplied with a sub-page and a home page of a website. In general, if an algorithm can correctly differentiate two 'sibling' pages, it should also be able to handle this test as well, but it is included for reference purposes.

For each of these scenarios, each algorithm was supplied with a couple of thousand randomly selected pairs of web pages from the test corpus. Based on the collected results, the average similarity and its standard deviation were calculated, together with average comparison times. The results are presented and discussed later in the paper.

2.2 The Algorithms

A number of algorithms were evaluated. They are briefly described below.

- *Levenshtein* is probably the most widely known edit distance metric, proposed in 1966 and published in [9].
- *Needleman-Wunch* is a similar insertion-deletion metric, but with a variable adjustment to the cost of the gap [13].
- *T*he *Jaro* distance metric comes from [8] and is meant to measure the transpositions between two strings, taking into account the relative distance between these transpositions. A modification to this algorithm proposed by *Winkler* [21] favors the characters at the start of the matching strings.

- *Q-Grams* are used in 'fuzzy' matching of two strings by comparing a number of short sub-strings (grams) that are found in both strings. More information on the q-grams and their applications can be found in [5].
- *Smith-Waterman* distance [17] was developed to identify optimal alignments between related DNA and protein sequences.
- *Soundex* is a search algorithm that compares words by the similarity of their sounds and originates from the 1917 US Patent by Robert C. Russell. Since the tokenization produces artificial data, it is not expected to work well, but was included for the reference purposes.
- *Jaccard* similarity was introduced by Jaccard in [7] to compare two finite sets of biological samples
- *Cosine* similarity is a normalized dot product between two vectors representing the compared documents.
- A *Sorensen* index, is a variation of Jaccard's formula [18].

3 Results of Algorithms Comparison

The results of the tests are presented in the following tables. Table 2 presents the aggregated results of algorithms applied to the three generated sets of page pairs (as explained above). In order to rank the algorithms for the purpose of identifying near-identical pages and separating them from other pages on the website, a quality metric Qs is introduced:

$$Q_s = (P_{\bar{x}} - S_{\bar{x}}) - (P_\sigma + S_\sigma) \tag{1}$$

Table 2 Page similarities values for various sets of web pages and different algorithms

	Same page comparisons		Against sibling pages		Against home page		Q_h	Q_s
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ		
Soundex	0.31	0.44	0.12	0.29	0.18	0.35	−0.66	−0.54
Jaro	0.74	0.36	0.36	0.32	0.26	0.23	−0.10	−0.30
Cosine	0.99	0.01	0.80	0.17	0.59	0.16	0.23	0.00
Jaro-Winkler	0.92	0.06	0.69	0.15	0.50	0.13	0.23	0.02
Sorensen	0.54	0.06	0.34	0.11	0.20	0.07	0.21	0.03
Needleman-Wunch	0.95	0.04	0.68	0.12	0.57	0.06	0.28	0.11
Jaccard	0.85	0.11	0.42	0.17	0.23	0.09	0.42	0.15
Levenshtein	0.92	0.06	0.51	0.20	0.26	0.13	0.47	0.15
Q-Grams	0.89	0.07	0.48	0.17	0.27	0.13	0.42	0.17
Smith-Waterman	0.87	0.10	0.36	0.20	0.27	0.14	0.36	0.21

where:

$P_{\bar{x}}$ is the average similarity between pairs of same-pages supplied to the algorithm. The similarity is within ⟨0–1⟩ range

Pσ is the standard deviation of similarity between pairs of same-pages supplied to the algorithm. The similarity is within ⟨0–1⟩ range

$S_{\bar{x}}$ is the average similarity between pairs of different sub-pages within one website ('siblings')

Sσ is the standard deviation of similarity between pairs of different sub-pages within one website ('siblings')

All the averages and standard deviations are calculated over subset of *n* pairs selected randomly from all the possible combinations (pairs) within the data set. Approximately 5000 page pairs were used for each test presented in this paper.

The metric Qs, identifies the quality of separation between the same and different pages of various websites. The higher the value of the metric, the better the algorithm is at identifying near-identical pages. Similarly, the Qh metric, which is constructed in the same way, represents the algorithms' ability to distinguish a sub-page from the website home page.

As can be seen in Table 2, Smith-Waterman algorithm performed best, and Soundex worst (the latter is expected, as this algorithm was designed for identifying phonetic similarity and is not meant for more generic use). Smith-Waterman also seems to be among the most sensitive to page difference, with one of the lowest similarity score for same pages comparisons. Still, the very low value for comparisons of sub-pages at the same site resulted in the best overall score.

The algorithms have also been tested with different pre-processing of the HTML to find out how it may impact their abilities. That included the removal of the tags from the HTML code, and encoding each token (see Fig. 1) with multiple bytes instead of one. The results are presented in Table 3.

Table 3 Results of adjustments to the HTML pre-processing

	Baseline		Excluding tags		Excluding tags and header		Double hash length	
	Q_h	Q_s	Q_h	Q_s	Q_h	Q_s	Q_h	Q_s
Soundex	−0.66	−0.54	−0.67	−0.51	−0.69	−0.52	−0.55	−0.57
Jaro	−0.10	-0.30	0.03	−0.10	0.02	−0.11	−0.07	−0.19
Cosine	0.23	0.00	0.32	0.17	0.32	0.17	0.17	0.01
Jaro-Winkler	0.23	0.02	0.31	0.19	0.21	0.07	0.15	0.09
Sorensen	0.21	0.03	0.21	0.09	0.21	0.08	0.20	0.04
Needleman-Wunch	0.28	0.11	0.23	0.14	0.23	0.14	0.28	0.12
Jaccard	0.42	0.15	0.38	0.24	0.38	0.24	0.41	0.18
Levenshtein	0.47	0.15	0.39	0.22	0.39	0.22	0.49	0.17
Q-Grams	0.42	0.17	0.29	0.20	0.29	0.19	0.42	0.16
Smith-Waterman	0.36	0.21	0.21	0.25	0.21	0.25	0.39	0.24

As can be seen in this table, encoding each piece of information into multiple bytes tokens does not noticeably affect the algorithm results. It does, however, impact the computing time significantly (see Table 4). Four byte encoding was also tested, but is not included in the table as it does not provide any new valuable information.

Excluding tags from the comparison provides significant benefits, as the Qs value is better for every single algorithm in this case. The order of the ranking of the algorithms changes slightly, but the top four remain the same (in a slightly different order).

Figure 2 illustrates the distribution of the similarity data for the Smith-Waterman algorithm for the scenario where tags are excluded from the computations. As can

Table 4 Average time of single page pair comparison (values in ms), for near-identical and different pair of pages

	Excluding tags (500 average length)		Baseline (1500 average length)		Double hash length (3000 average)	
	Similar	Different	Similar	Different	Similar	Different
Soundex	6	6	8	8	9	8
Jaro	7	7	16	14	38	29
Cosine	8	9	30	38	92	136
Jaro-Winkler	6	6	8	8	9	10
Sorensen	7	7	14	12	30	22
Needleman-Wunch	7	7	22	19	65	53
Jaccard	7	7	15	13	31	23
Levenshtein	8	7	21	17	57	46
Q-Grams	8	8	27	33	72	102
Smith-Waterman	7	7	27	22	85	63

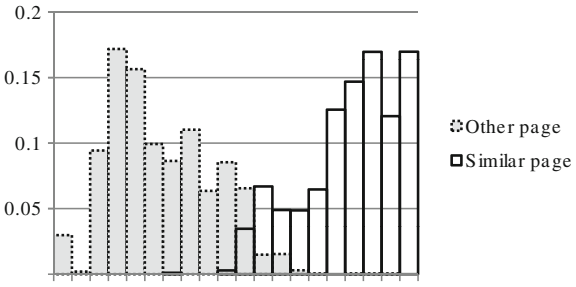

Fig. 2 Distribution of page similarity values for Smith-Waterman algorithm

be seen in this figure, the separation of results between the same pages and other sub-pages is quite good, with around 0.07 (7 %) of aggregated cross-coverage.

3.1 Computing Time

Finally, the efficiency of the algorithms in terms of processing time was assessed. Table 4 presents the list of average comparison times for each algorithm.

Obviously, the times depend on the computing platform processing power, but since all tests were performed on the same platform (Intel i7 2.6 GHz CPU, single thread), the relative comparison is still valid. It can be assumed that about 5 ms is needed for the pre-processing (HTML parsing, hashing, etc.). The impact of the comparison algorithms for short strings, approximately 500 bytes long is negligible. It does however grow exponentially for most algorithms with the increase of compared hashed lengths. The times do not seem significant, but with the target application of automated scanning and classification of retrieved website's pages, the number of comparisons can easily reach millions, and in this case the decrease of comparison time due to the exclusion of tags from the task, provide noticeable benefits.

4 Conclusions and Future Work

The objective of the research was to identify the best algorithms for effective page comparison and identification of (near) identical web pages. This functionality is needed for the development of new, automated security scanners for dynamic web applications. Such scanners, especially these utilizing *fuzzing* of the parameters, generate a large number of requests to the web application (the more the better), and will need to execute millions of page comparisons. Because of this, the speed of single comparison is crucial and therefore text similarity metrics were tried in preference to other, more computationally complex methods (e.g. based on DOM tree similarity).

A number of algorithms were tested, which were supplied sets of tokenized version of both similar and different pairs of web pages. Each algorithm was used to calculate a normalized value of similarity for each supplied pair of web pages. The higher and more consistent (i.e. lower standard deviation) values for pairs of similar pages, and at the same time, lower and consistent similarity values for pairs of different pages, the better suited the algorithm is for the specified task. This was formally specified as an algorithm's separation quality and used to rank the algorithms.

It was revealed, that several algorithms could be considered for the specified task, with *Smith-Waterman* scoring the best results and *Jaccard* and *Levenshtein* following it closely. Not unexpectedly, this proves that the choice of the metric used

must be determined by the task it is employed for, as e.g. *Jaro* metrics was better at comparing short advertisements [22].

In order to further improve the results and the separation quality for different web pages, it is probably necessary to go beyond the use of a simple metrics. As suggested in [3], hybrid methods, which utilize more than one algorithm, may perform better—but this will likely result in longer processing times, which is crucial for the specified task. Another option might me to use additional methods only to a subset of comparisons when the decision is uncertain. In order to further improve the quality of separation and increase the certainty of identifying near-identical dynamically generated pages, it is also planned to investigate machine learning techniques that will adapt to the specific site being tested.

References

1. Alpuente, M., Romero, D.: A visual technique for web pages comparison. Electr. Notes Theor. Comput. Sci. **235**, 3–18 (2009)
2. Clayton R.: String Metrics Library: https://github.com/rclayton/StringSimilarity
3. Cohen W., Ravikumar P., Fienberg S.: A comparison of string metrics for matching names and records. In: KDD Workshop on Data Cleaning, Vol. 3 (2003)
4. Fu, A.Y., Wenyin, L., Deng, X.: Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). IEEE Trans. Dependable Sec. Comput. **3** (4), 301–311 (2006)
5. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Pietarinen, L., Srivastava, D.: Using q-grams in a DBMS for approximate string processing. IEEE Data Eng. Bull. **24**(4), 28–34 (2001)
6. Henzinger, M.: Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 284–291. ACM (2006)
7. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytol. **11**, 37–50 (1912)
8. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J. Am. Statist. Assoc. **84**(406), 414–420 (1989)
9. Levenshtein, V.: Binary codes capable of correcting deletions and insertions and reversals. Soviet Physics Doklady **10**(8), 707–710 (1966)
10. Lucca, G.D., Penta, M.D., Fasolino, A.: An approach to identify duplicated web pages. In: Proceedings of International Computer Software and Applications Conference (COMPSAC), pp. 481–486 (2002)
11. Lukashenko, R., Graudina, V., Grundspenkis, J.: Computer-based plagiarism detection methods and tools: an overview. In: CompSysTech. ACM International Conference Proceeding Series, Vol. 285, p. 40. ACM (2007)
12. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
13. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**(3), 443–453 (1970)
14. Pera, M.S., Ng, Y.K.: Identifying spam web pages based on content similarity. In: ICCSA (2), Vol. 5073, pp. 204–219. Lecture Notes in Computer Science. Springer, Berlin (2008)
15. Rosiello, A.P., Kirda, E., Kruegel, C., Ferrandi, F.: A layout-similarity based approach for detecting phishing pages. In: Security and Privacy in Communications Networks and the Workshops, pp. 454–463. IEEE (2007)

16. SimMetrics, a Similarity Metric Library: http://sourceforge.net/projects/simmetrics/
17. Smith, T., Waterman, M.: Identification of common molecular subsequences. J. Mol. Biol. **147** (1), 195–197 (1981)
18. Sorensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons". Kongelige Danske Videnskabernes Selskab **5**(4), 1–34 (1948)
19. Symantec Internet Security Threat Report, Vol. 20, http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=istr-20 (2015)
20. Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., Deng, X.: Detection of phishing webpages based on visual similarity. In: 14th international conference on World Wide Web, pp. 1060–1061. ACM (2005)
21. Winkler, W.E.: String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: Proceedings of the Section on Survey Research, pp. 354–359 (1990)
22. Zachara M., Piskor-Ignatowicz C.: Comparison of string metrics effectiveness for the purpose of estimating the number of unique job offers. PAR (11), pp. 213–216, PIAP (2011)

Distributed Web Server's Data Performance Processing with Application of Spatial Econometrics Models

Leszek Borzemski and Anna Kamińska-Chuchmała

Abstract Predicting the download time of Web distributed resources is an important but challenging problem in the worldwide public Internet. In our recent work we focus on spatial-based methods. We propose to use spatial econometrics methods to predict Web server's performance. Three spatial regression models have been studied: Classical Regression Model (CRM), Spatial Lag Model (SLM) and Spatial Error Model (SEM). We used the real-life dataset obtained in active experiments performed by our Virtual Multiagent Internet Measurement System (VMWING), which monitored web transactions issued by VMWING's agent located in Wrocław, Poland and targeting Web servers in Europe. We also compared studied econometrics methods with geostatistical methods which were analyzed in our previous papers.

Keywords Web server's performance prediction · Spatio-temporal data mining · Spatial econometrics · Regression models

1 Introduction

Nowadays we live inside a mobile Internet info sphere which among others include the Web with its permanent performance problems. The World Wide Web is a distributed Internet based system, which provides a many type of services and resources (e.g. data, files, media) for users. We need to have Web more reliable and predictable what is especially needed in the context of 3G/4G distributed mobile communications where the Web resource download time plays a major impact on users' experience. One may consider many cases where there is a need to choose

L. Borzemski (✉) · A. Kamińska-Chuchmała
Faculty of Computer Science and Management, Wrocław University of Technology, Wrocław, Poland
e-mail: leszek.borzemski@pwr.edu.pl

A. Kamińska-Chuchmała
e-mail: anna.kaminska-chuchmala@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_4

which Web service to use. Then a prediction which server provides the best service is crucial. However predicting the download time of Web distributed resources a challenging problem in the worldwide public Internet. Various approaches have been employed, including these presented in our papers, in order to construct predictors which could be classified as formula-based, history-based, class-based or value-based. These methods, regardless of their structure allow to evaluate the performance with time for a given Web server. For example, in a class and history-based data mining method we are able to predict classes of performance of a Web server by means of a clustering and decision tree methods [1]. Another example is [2] where we demonstrated employed the Transform Regression (TR) algorithm to predict file transfer time. We considered the files that were downloaded from several Web servers using HTTP protocol. We identified parameters having impact on file download time and used them as inputs in Transform Regression based prediction algorithm. We compared also the results of TR-based prediction with the moving averages methods.

Because the Web is a geographically distributed system hence naturally arose research question whether it should consider methods of spatial prediction. Therefore in our recent work we have focused on various spatial-based methods, including geostatistics and spatial econometrics. The former works are presented in various our papers listed in the Reference section whereas the latter is studied here.

In this research we propose to use spatial econometrics methods to predict Web server's performance. Three spatial regression models have been studied: Classical Regression Model (CRM), Spatial Lag Model (SLM) and Spatial Error Model (SEM). We used the real-life dataset obtained in active experiments performed by our Virtual Multiagent Internet Measurement System (VMWING), which monitored web transactions issued by VMWING's agent located in Wrocław, Poland and targeting Web servers in Europe. We also compared studied econometrics methods with the geostatistical methods which were analyzed in our previous research.

The rest of the paper is organized as follows. Section 2 introduces our approach to Web prediction. Section 3 defines spatial econometrics models. Section 4 describes active performance measurement experiment and data analysis performed. Section 5 shows the results of the performance using spatial econometrics models. Section 6 presents the spatial econometrics methods vs. geostatistical methods. After discussing related work in Sect. 7 we conclude in Sect. 8.

2 Web Performance Prediction Approach

The Web uses HyperText Transfer Protocol (HTTP) application protocol designed for providing a data communication between Web clients and Web servers storing Web resources. Basically, data transfer is done by means of Transmission Control Protocol (TCP) which uses Internet Protocol (IP) to transmit packets through the network. Because of several reasons current Internet communication is only the

best-effort and does not provide (without expansions) any predictable communication performance quality to the end users.

However in an everyday use of Web, the users need the predictability in Web performance. The predicted performance is needed, for example in the optimal selection of Web servers to download required resource is it is stored in different Internet nodes. This can be especially important in current 3G/4G based mobile systems as they have several internal characteristics that may have bad influence on Web-based communication.

However, examining the performance of the Web server, we should ask ourselves how we will evaluate it. Well, not having possibilities of direct measurement of performance of web server, we need to apply some indirect measurement. In addition, we must remember that we want to investigate the performance—and in the future to predict it—from the point of view of some end user. What's more, you should consider factors that can affect Web server performance. For example, in [2] we analyzed a number of such factors/parameters that could be measured during file transfer or calculated. Their values characterize conditions of each data transfer. Our experiment on PlanetLab showed that it is possible to study even 125 features describing data transfer properties [3]. Such a collection is not to comprehend in a simple way. Moreover most of these factors cannot be measured by an ordinary Web client software, that is a browser.

In this research we focus only on geographic distance and time of day - the link conditions can differ on the time of day, at some hours the traffic aggregated is less and helps us receive better times than at rush hours. The measure of Web performance is the time to download a given Web resource—such performance measure may be used as we download the same resource from different Web servers in various geographic locations. In our previous work we found several times that the geographical localizations has a major influence on Web performance. This also is confirmed in other research, what is discussed in related work section of this paper. This is why we are interested in spatial-based prediction approaches to Web performance.

3 Spatial Econometrics Models

In 1970 Waldo Tobler started of spatial econometrics by stating: "Everything is related to everything else, but near things are more related than distant things" [4]. This sentence was formulated as first law of geography, and four years later Jean Paelinck introduced the notion of spatial econometrics. Moreover Paelinck together with Klaassen wrote about the five characteristic features of spatial econometric methods [5]: the asymmetry in spatial relations, the importance of explanatory factors located in other spaces, the role of spatial interdependence in spatial models, and differentiation between ex post and ex ante interaction and explicit modeling of space.

Luc Anselin proposed division of spatial econometrics into two groups [6]: spatial dependence (or autocorrelation) and spatial heterogeneity.

Generally spatial econometrics (regression) models include relationships between variables and their neighboring values. These models were created by joined econometric, economic, and geography methods. The area of application of spatial econometric models are very wide, e.g., Spatial Lag Model (SLM) and Spatial Error Model (SEM) were used to modeling the spatial dependence of state travel on all twelve types of highway for the U.S. over the period 1980–2008 [7] or there were used to spatial pattern analysis of economic growth of JingJinJi metropolitan region over the 2001–2006 periods [8].

The next three subsection described spatial econometrics models used in which will be used for spatial prediction of Web system performance (WSP).

3.1 Classical Regression Model (CRM)

A standard model of linear regression (Classic Regression Model CRM) has the following form:

$$y = X\beta + \varepsilon \tag{1}$$

where y is a vector of observations on the dependent variable, X is a matrix of observations on the independent variables, β is model's coefficient and ε is a random component. Such model is not always sufficient in practice, hence it uses hybrid models and models with weights.

Spatial weight matrices W are formed on the basis of the distance matrix or neighbourhood matrix and treated as independent variables in econometric model. Interaction strength between two points i and j depends on the distance between them. For larger distances between two points, the interaction strength should be smaller. This can be interpreted as the presence and strength of a link between points (the observations) in a network representation that matches the spatial weights structure. Therefore, in weight matrix W, computed on the basis of the distances dij, particular elements are in general inverse functions or exponential-inverse functions of distances:

$$w_{ij} = d_{ij}^{-\alpha}, w_{ij} = e^{-\alpha d_{ij}}, \tag{2}$$

where α is predetermined parameter.

3.2 Spatial Lag Model (SLM)

In contrast to CRM, which uses eigenvalues of the weights matrix W, Spatial Lag Model (also known as Spatial Autoregressive Model (SAR)) is well suited to the

estimation in situations with very large data sets. In matrix notation SLM, could be written:

$$y = \varrho Wy + X\beta + \varepsilon \tag{3}$$

where ρ is autoregressive parameter and $\varepsilon \sim N(0, \sigma^2 I)$.

3.3 Spatial Error Model (SEM)

Spatial Error Model (SEM) does not require a theoretical model for spatial interaction like SLM, but is a special case of non-spherical error covariance matrix. SEM is a model with spatial linear autocorrelation of random component:

$$y = X\beta + \xi \tag{4}$$

$$\xi = \lambda w\xi + \varepsilon \tag{5}$$

where ξ is a random component, λ is a parameter of model and $\varepsilon \sim N(0, \sigma^2 I)$.

4 Active Experiment and Data Analysis

Database for research were created on basis an active experiment Virtual Multiagent Web pING (VMWING). This experiment was performed in authors Distributed Computer Systems Division at Wrocław University of Technology and is extension of MWING experiment [9, 10]. The idea of VMWING is to gather information about Web server performance in the network. This is distributed experiment on one hundred European servers from which agent in Wrocław is downloading a copy of the same resource: server install image for PC (Intel x86) computers (zsync metafile). VMWING relies on Web transactions with using HTTP protocol. The size of downloading resource is 1.3 MB. The database include the information about the total downloading time of ubuntu-13.04-server-i386.iso.zsync file, server's geographical location which the agent targeted, and the time stamp of taking a measurement. The experiment were taken between 4th and 24th of October 2013, every day at the same time: 06:00 am, 12:00 pm, 6:00 pm and 12:00 am. The data from this experiment were used to spatial prediction with using geostatistical methods: Turing Bands [11] and Ordinary Kriging [12]. In Sect. 6 results obtained with using spatial econometrics models will be compared with geostatistical methods results.

In Table 1 the elementary statistics of web servers performance history from database for particular hours are presented.

Analyzing the data in Table 1, it could be claimed that there is large difference between minimum and maximum values equal more than 35 s, which indicated on dispersion in the data. Additionally, average values are equal about 6 s for four considered hours, which is quite small in compared to whole range. Moreover the highest variability (above 100 %), skewness, and kurtosis coefficients for midnight hour may point to less performance of Web servers at midnight or that the part of servers in network are disabled at night and then communications is slower.

In Fig. 1 cartogram of total download time from resource for the 4th day of October 2013 at midnight is presented. As could be seen, there are 6 servers where waiting time on file is equal about half of minute. The changeability of download time is varied in adjacent servers.

Bivariate Moran's *I* statistics between exemplary days 4th and 5th October 2013 for 06 pm are presented in Fig. 2. Presented Moran's *I* statistics of input data is very

Table 1 Elementary statistical parameters of download times of resources from servers during period 4–24.10.2013 [9]

Statistical parameters	6:00 am	12:00 am	6:00 pm	12:00 pm
Minimum value X_{min} (s)	0.00	0.09	0.08	0.08
Maximum value X_{max} (s)	35.95	35.95	35.91	35.99
Average value X (s)	6.14	6.26	6.58	5.86
Standard deviation S (s)	6.19	6.14	6.44	5.96
Variability coefficient V (%)	100.81	98.08	97.87	101.71
Skewness coefficient G	2.98	2.87	2.84	3.12
Kurtosis coefficient K	12.57	11.88	11.38	13.89

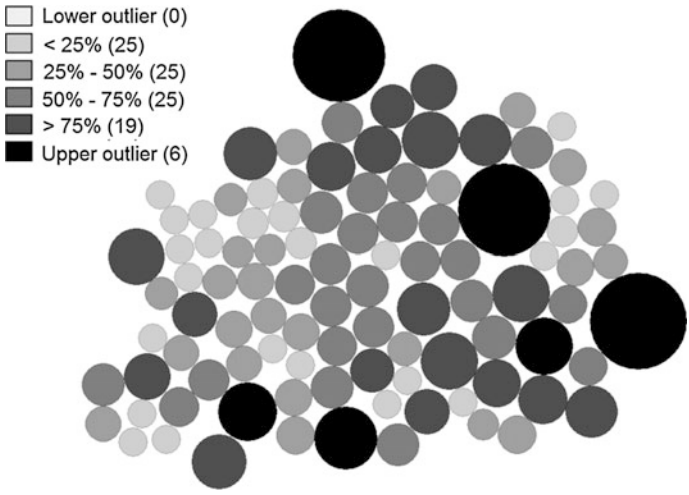

Fig. 1 Cartogram of total download time for 4th October 2013 at 12 pm

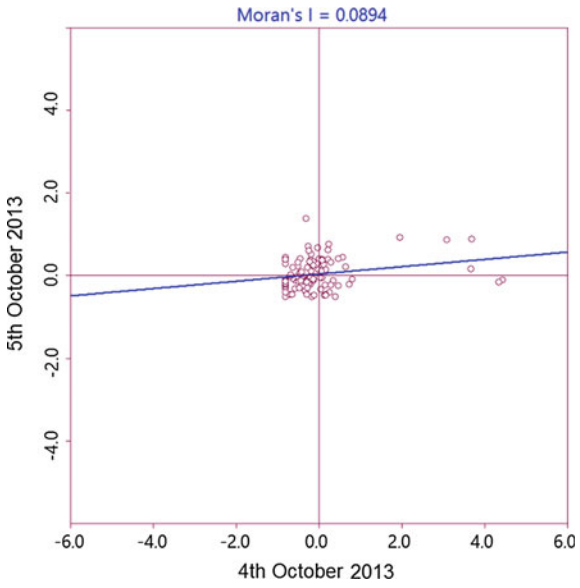

Fig. 2 Bivariate Moran's *I* statistics between 4th and 5th October 2013 for 06:00 pm

small—only 0.089. During research autocorrelation between other days is similarly close to zero. These results from Moran's statistics indicates the large heterogeneity among Web servers for particularly hours and days in considered database.

5 Prediction of Web Server's Performance Calculated with the Spatial Econometrics Models

Three types of models were used to predict the total download time of file from Web servers: CRM, SLM and SEM. Prediction was calculated with one week advance, i.e. between 25th and 31st October 2013 and for four hours during every day: 06:00 am, 12:00 pm, 6:00 pm and 12:00 am. Models assumed, that dependent variable was a day, which it will be predict and independent variables were the days of history in database discussed in Sect. 4. To create matrix weights Euclidean distance was used. Threshold distance was equal 8.65° and variables for x and y coordinates was assumed as centroids. All predictions and data analysis were performed with using GeoDa software [13].

Results of prediction for 31st October 2013 total download file for exemplary two servers are presented in Table 2. First presented server is located in Centro Informático Científico de Andalucía in Seville (Spain) and the ubuntu-13.04-server-i386.iso.zsync file was downloaded from url: ubuntu.cica.es. Second server is located in Academic Computer Centre in Gdansk—TASK affiliated to Gdańsk University of Technology in Gdansk (Poland) and the file was downloaded from url: ubuntu.task.gda.pl. Analyzing the accuracy of prediction with using three

spatial econometric models it could be claimed that the most difficult prediction was in the morning. In Seville the smallest error of prediction calculated with SLM was equal 34.74 % and in Gdańsk 13.99 % also with using SLM. The problem with accuracy for this morning hour it follows that the previous days download time was equal about 5 s, but last day of October was equal 29.51 s. This fact only confirmed the conclusions from analysis of input data, that is characterized changeability of examined process and autocorrelation close to zero between data. However for other hours (except noon in Seville) the accuracy of prediction is very good, in some cases error is less than 1 %.

To confirm the reason of small accuracy of performance prediction of server in Seville at 12 am, the history of real download time during October in Fig. 3 is presented. There is visible very high dispersion between data. For three days 15th, 20th and 28th of October the time to download Ubuntu file from servers were equal above 30 s while most days' time was below 10 s.

Table 2 Average prediction error ex post for exemplary two Web servers in Europe for prediction for 31st October 2013, compared models

Spatial econometric models	6:00 am	12:00 am	6:00 pm	12:00 pm
Server in Seville, Spain				
Error ex post (%) for CRM	36.07	39.58	1.71	1.03
Error ex post (%) for SLM	34.74	39.54	2.29	4.81
Error ex post (%) for SEM	37.42	37.57	1.45	8.35
Server in Gdansk, Poland				
Error ex post (%) for CRM	18.85	2.41	0.03	3.28
Error ex post (%) for SLM	13.99	2.52	1.06	5.38
Error ex post (%) for SEM	28.30	7.68	0.71	2.71

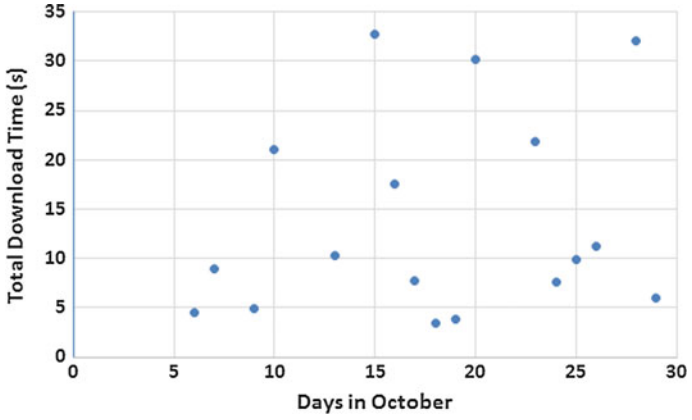

Fig. 3 Total download time of file from server in Seville (Spain) during period 4th October to 31st October 2013 at 12:00 am

6 Spatial Econometrics Methods Versus Geostatistical Methods

Usually, the majority of the research in spatial data analysis can be divided into two branches: the model-driven approach and the data-driven approach. The spatial econometric methods use the model-driven approach where the crucial issue is to take appropriate model which will takes into account relationship between servers with using spatial autocorrelation. These methods such as CRM, SLM, SEM, could analyze spatial heterogeneity of the examined process. On the other hand, there are some restrictions—in model we must know approximate value of dependent variable. If we want to only verify correctness of assumed model, that is the best appropriate way. Moreover this kind of prediction we can made iteratively, only one step (in our case one day) ahead. As a result we obtain admittedly spatial prediction for this considered Web servers, but only for them.

Authors used geostatistical simulation and estimation methods in their previous research e.g., presented in [14–17] and also on the same database considered in this paper [11, 12]. This spatial statistics–geostatistics uses the data-driven approach, where the search ellipsoid is used to consider neighbourhood of given Web server. In addition, the studies are performed with the 3D simulation grid and as a result these methods could give information about performance not only for considered servers, but for a whole investigated area (please see exemplary Fig. 4 from prediction results on basis the same considered in this paper database). This spatial prediction based on 3D directional variogram gives us results as much ahead as it wants without any hints about the predicted value.

In [16] authors compare simulation (Turning Bands—TB and Sequential Gaussian Simulation—SGS) with estimation methods (Simple Kriging) by performing spatial prediction on the same database. The results give information about advantage simulation over estimation methods. On the other hand it is difficult

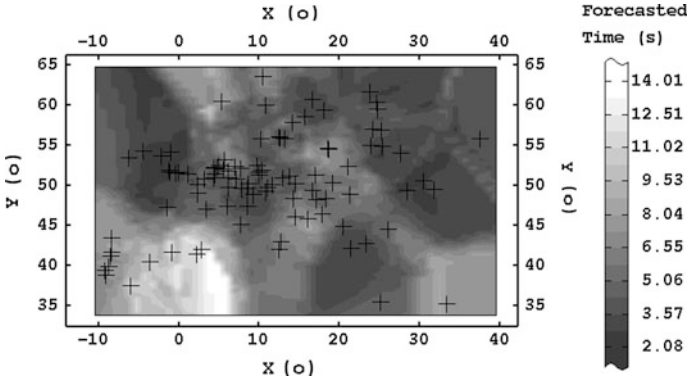

Fig. 4 Raster map of total download time of file in European Web servers in 31st October for 06:00 pm calculated turning bands method [9]

compare econometrics and geostatistical methods in this context, because these give different quality information about such prediction: econometric about autocorrelations between spatial data (Web servers) and simulation about Web server performance for a whole considered area. However, there were the attempts to compare this methods in other domain—to analyzing the crime rate in Columbus (Ohio) [18]. Author showed advantage of geostatistical estimation methods over spatial econometric methods. Mean absolute error was higher at least 2 %.

7 Related Work

Reference [1] introduced the new data mining driven performance prediction model called Web Performance Mining (WPM). We applied a similar classification-based prediction approach road map to forecast RTT states for IP networks [19].

Research on predicting file download time and bandwidth or link capacity is mostly based on an active measurements where a probe file, size of which is much less than of the file to be transferred, to gather some communication path properties which later are used in the prediction. Another method which cannot be implemented in each situation is based of continuous monitoring using a small probe as above or a specific file which is to be transferred. These methods are known as Variable Packet Size (VPS) probing [20, 21], Packet Pair/Train Dispersion (PPTD), Self-Loading Periodic Streams (SLoPS), and Trains of Packet Pairs (TOPP). They send packets of a given size in a scheduled manner and calculate of the predicted bandwidth or link capacity by using inter-arrival times of the returned packets. Another approach [22] is using some knowledge of TCP flow patterns to predict future TCP throughput. The throughput is then measured in the intervals of Round Trip Time (RTT) what constitutes the time series of throughput values that is analyzed to make a prediction.

Active measurements and history-based data are used as inputs to autoregressive (AR), autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models [23].

As performance prediction is one of the key issues of Internet and Web various researches considered the Internet performance through different performance measures like the transmission delay (RTT), data throughput (TCP throughput). Web performance is experience by the Web page total page load time or individual Web resources download times [24–26].

8 Conclusion

The approach we studied gave acceptable results as far as the study approach is concerned. We used real-life measurement data concerning Web performance of a set of Web servers. We performed prediction analysis by means of econometrics

methods and compared this approach to other spatial based method, namely geostatistics. We claim that presented econometrics methods to prediction Web server performance could be helpful in spatial analysis of Internet characteristics.

The future research includes an active measurement experiment performed in a new manner to gain new measurement attributes. Also other methods from the domain of spatial statistics to predict Web performance will be studied.

References

1. Borzemski, L.: The use data mining to predict Web performance. Cyber. Syst. **37**(6), 587–608 (2006)
2. Borzemski, L., Starczewski, G.: Application of transfer regression to TCP throughput prediction. First Asian Conference on Intelligent Information and Database Systems (ACIIDS), pp. 28–33, 1–3 April 2009
3. Borzemski, L., Rodkiewicz, M., Starczewski, G.: Internet distance measures in goodput performance prediction. In: HET-NETs, pp. 153 – 166 (2010)
4. Tobler, W.: A computer model simulating urban growth in the detroit region. Econ. Geogr. **46**(2), 236 (1970)
5. Paelinck, J.H.P., Klaassen, L.H.: Spatial Econometrics. Saxon House, Farnborough (1979)
6. Anselin, L.: Spatial Econometrics: Methods and Models. Kluwer Academic Publishers, Dordrecht (1988)
7. Glass, A.J., Kenjegalieva, K.: The economic case for the spatial error model with an application to state vehicle usage in the U.S. Rice University (2012)
8. Lian, J, Li, X., Gong, H., Wang, Y., Sun, Y.: The spatial pattern analysis of economic growth of JingJinJi Metropolitan Region. 18th International Conference on Geoinformatics, pp. 1–5 (2010)
9. Borzemski, L., Cichocki, L., Kliber, M., Fraś, M., Nowak, Z.: MWING: A Multiagent System for Web Site Measurements. LNCS, vol. 4496, pp. 278–287 (2007)
10. Borzemski, L., Cichocki, L., Kliber M.: Architecture of Multiagent Internet Measurement System MWING Release 2. LNCS, vol. 5559, pp. 410–419 (2009)
11. Kamińska-Chuchmała, A.: Forecast of Internet network loads as a proposition to improving efficiency in communication of smart metering. Rynek Energii **2**(111), 127–131 (2014)
12. Kamińska-Chuchmała, A.: Spatial Internet traffic load forecasting with using estimation method. Procedia Comput. Sci. **35**, 290–298 (2014)
13. https://geodacenter.asu.edu
14. Borzemski, L., Kamińska-Chuchmała, A.: Distributed web systems performance forecasting using turning bands method. IEEE Trans. Industr. Inf. **9**(1), 254–261 (2013)
15. Borzemski, L., Kamińska-Chuchmała, A.: Client-perceived web performance knowledge discovery through turning bands method. Cybern. Syst. Int. J. **43**(4), 354–368 (2012)
16. Borzemski, L., Kamińska-Chuchmała, A.: Web performance forecasting with kriging method. In: Contemporary Challenges and Solutions in Applied Artificial Intelligence Studies in Computational Intelligence, vol. 489, pp. 149–154. Springer, Berlin (2013)
17. Borzemski, L., Kamińska-Chuchmała, A.: Spatio-temporal Web Performance Forecasting with Sequential Gaussian Simulation Method, Communications in Computer and Information Science, vol. 291, pp. 111–119. Springer, Berlin (2012)
18. Calderon, G.F.-A.: Spatial regression analysis vs. kriging methods for spatial estimation. Int. Adv. Econ. Res. **15**(1), 44–58 (2009)
19. Borzemski, L.: Internet path behavior prediction via data mining: conceptual framework and case study. J. Univ. Comp. Sci. **13**(2), 287–316 (2007)

20. Claffy, K., Dovrolis, C., Murray, M.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Netw. **17**(6), 27–35 (2003)
21. Dovrolis, C.: End-to-end available bandwidth estimation. In: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 265–276 (2005)
22. Huang, T., Subhlok, J.: Fast pattern-based throughput prediction for TCP bulk transfers. In: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05), pp. 410–417 (2005)
23. Karrer, R.P.: TCP prediction for adaptive applications. In: Proceedings of the 32nd IEEE Conference on Local Computer Networks, pp. 989–996 (2007)
24. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to TCP throughput prediction. IEEE/ACM Trans. Networking **18**(4), 1026–1039 (2010)
25. Yin, D., Yildirim, E., Kulasekaran, S., Ross, B., Kosar, T.: A data throughput prediction and optimization service for widely distributed many-task computing. IEEE Trans. Parallel Distrib. Syst. **22**(6), 899–909 (2011)
26. Güngör, V.C., Sahin, D., Kocak, T., Ergüt, S., Buccella, C., Ceceti, C., Hancke, G.P.: Smart grid technologies: communication technologies and standards. IEEE Trans. Ind. Inform **7**(4), 529–539 (2011)

Key Requirements and New Architecture for Context-Aware Web-Based Device-Independent Multi-device Applications

Jacek Chmielewski and Martin Lasak

Abstract Cross-platform mobile and web applications become increasingly popular among users. As users begin to use multiple devices, their behavior patterns start to migrate from single-device to multi-device scenarios, where multiple devices are used sequentially or complementary to interact with a user and gather context information. Application developers should follow these trends and focus on multi-device aspects of their applications but efficient and cost effective application development requires proven frameworks and architectures, which are missing in this field. In this paper we identify key aspects of context-aware multi-device device-independent applications that should be supported by such tools and we propose a new application architecture. The presented architecture and conclusions are based on experimental implementations of a sample context-aware device-independent multi-device application called LifeLog.

Keywords Device-independent applications · Multi-device applications · Context-aware applications · Multi-device framework · Cross-platform applications · Mobile web applications · Webinos · Device-Independent Architecture

1 Introduction

The number of devices connected to the Internet is constantly growing. Cisco reports [1] that in 2013 the number of mobile-connected devices exceeded the world's population. The diversity of these devices enables new use cases. Users

J. Chmielewski (✉)
Poznan University of Economics, Poznan, Poland
e-mail: chmielewski@kti.ue.poznan.pl

M. Lasak
Fraunhofer FOKUS, Berlin, Germany
e-mail: martin.lasak@fokus.fraunhofer.de

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_5

start to use multiple different devices for the same activity. For example, it is more convenient to check weather forecast on a large TV while at home, but users will do the same on their smartphone while on the go. According to The New Multi-screen World Study done by Google [2], 90 % of users use multiple devices sequentially to accomplish a task over time. Therefore, the main focus of mobile application developers is on cross-platform capabilities of their applications that ensure proper application execution on many devices while maintaining reasonable development costs. However, the same study shows that sequential usage is accompanied by simultaneous usage, where users are using more than one device at the same time. Currently, 22 % of simultaneous usage is complementary (related activities on different devices) and it can be expected that this number will increase along with the growing number of devices that surround each user. The common need will be not only to switch from one device to another to continue a task, but to employ a number of devices at the same time for the same task. Therefore, the focus of application developers should shift to multi-device aspects of their applications. But currently, it is still unclear what the critical aspects of context-aware device-independent multi-device applications really are and how to support them.

We address these issues by proposing key requirements that have to be satisfied to successfully develop a context-aware device-independent multi-device application and by researching how these requirements can be satisfied by the Device-Independent Architecture (DIA) [3] and by the multi-device webinos application architecture [4]. Main contributions of this paper are the following:

- Identification of key aspects of context-aware device-independent multi-device applications.
- Verification of two application architectures (DIA and webinos).
- Definition of a combined architecture that satisfies all defined requirements.

The verification is based on a case study of an electronic diary application called LifeLog, which is designed to include all defined requirements. The LifeLog application forms a common ground on which we can compare both analyzed architectures and identify their shortcomings. The implementation of these two LifeLog application variants provides valuable insights, which are used as guidelines for an application architecture that combines the best concepts and enables development of truly context-aware device-independent multi-device applications.

The paper is composed of six sections. Section 2 contains background information related to concepts of device-independence and multi-device usage. In Sect. 3, the key multi-device application requirements are defined. Section 4 provides description of architectures verification, along with an overview of DIA and webinos architectures. Section 5 includes a discussion on the results of the verification and a definition of the new multi-device application architecture. Section 6 concludes this research work and points out directions for future research.

2 Background

Preparation of a device-independent multi-device application (DIMDApp) requires tackling two interrelated problems. The first is about making an application independent of device capabilities. The second is about enabling multi-device usage scenarios.

In this paper, a device-independent application is an end-user application that offers its functions to a user regardless of a device used by the user. The device independence problem is known in the distributed systems domain as technology transparency [5] and has been addressed in many ubicomp environments such as Gaia [6] or iRos [7]. However, with the advent of diverse personal devices such as smartphones and tablets, it became an immanent issue for all mobile application developers, not only developers of distributed systems. If developers want to provide their applications on a wide range of different platforms then they have to focus on building cross-platform applications—i.e., applications that may operate on different hardware or software platforms without the need to rewrite the application code. Following such definition of a cross-platform application, it is possible to distinguish two classes of applications: hardware-independent and OS-independent applications.

Hardware-independent applications are written for a specific software platform (e.g., an operating system (OS)) and use hardware abstraction layer (HAL) provided by the software platform to access hardware features in a universal manner. This class of applications includes all native applications written for operating systems such as Google Android or Apple iOS. The HAL separates an application from device hardware internals, but the application still has to accommodate different form factors of devices used by users [8]. For example, a usability of a Graphical User Interface (GUI) prepared for a large screen (e.g., for a tablet) may be significantly deteriorated when presented on a small screen (e.g., on a smartphone) if no countermeasures are considered [9]. Another problem is the reach and version fragmentation of operating systems [10]. An application developed for a specific OS will have its portability limited to devices running this OS. This issue is even more significant if the operating system HAL API changes considerably between OS versions. In such a case, the application portability can be limited even more—only to devices running a specific version of an OS. An example of these issues is the Android OS fragmentation [11].

To overcome these problems developers avoid building OS-native applications and use software execution platforms that add another abstraction layer, on top of the OS. Taking into account strategies of execution platform runtime distribution, it is possible to distinguish two general types of execution platforms: universal execution platforms and custom execution platforms. Universal execution platforms have runtime distributed independently of applications, while custom execution platforms have the runtime embedded within the application package.

Examples of universal execution platforms include: Web, Java, and Flash. The Web platform is an open platform with publicly available specifications and

multiple implementations of its runtime, which in this case is a Web browser. The Java platform is owned by Oracle, but its specification and runtime implementation are open. Apart from the original Java Runtime Environment there exist multiple implementations of the Java runtime. The Flash platform is another proprietary platform (owned by Adobe). It has a partially open specification. However, despite efforts of projects Gnashm, swfdec, and Lightspark, the only fully reliable runtime is the original Adobe Flash Player. The intention of open specifications and multiple implementations of a runtime is to solve the reach problem that plagues native applications. But, in practice, this goal is not attainable. Small differences in implementations of a runtime (e.g., in Web browsers [12]) or changes between different versions of a runtime (e.g., in Flash Player) cause fragmentation problems similar to OS fragmentation. According to a cross-platform tools benchmarking study done by research2guidance [13], the fragmentation is invariably one of the top barriers to widespread adoption of cross-platform tools. Therefore, developers have to limit their code to the lowest common denominator [14] to make sure an application will work properly on all variants of a runtime or enhance their code by checks and runtime dependent passages.

Custom execution platforms, which have its runtime distributed with the application itself, provide better execution environment in terms of runtime implementation consistency. The number of custom execution platforms is significant. The Cross-platform Developer Tools report prepared by VisionMobile [15] presents a list of 100 tools that use technology approaches ranging from custom runtimes and source code translators, to app factories and Web-to-native app wrappers. The most popular cross-platform development tools for smartphone applications, presented and analyzed by Ohrt and Turau [16], cover all popular mobile operating systems and effectively solve the reach and fragmentation problems. Unfortunately, it is true only for the most popular platforms. More exotic platforms, with smaller user-base, are harder to reach because they are not economically viable targets for providers of custom execution platforms.

It is important to notice, that both types of execution platforms have issues with performance and functionality. The extra abstraction layer of an execution platform brings its own API on top of the API provided by the OS, which often extends the processing pipeline and makes applications less responsive. Moreover, an API functionality existing on one system might be missing on another, effectively shrinking the capability set of the unified abstraction layer.

The device independence topic, analyzed in this paper, is similar to the cross-platform topic, but approaches the problem from a different angle. Cross-platform developers focus on seamless application execution on any device, while the device independence topic is more about seamless application usage by end-users across all their devices. So, the device independence involves all the aspects included in the cross-platform development topic, plus issues such as useful application user interfaces on various screens and migration of application functionality (and data) from one device to another.

As already stated, the second aspect of DIMDApps is about enabling multi-device usage scenarios. The term multi-device application is sometimes

misused to describe cross-platform applications—i.e., applications that may be executed on multiple different devices. In this paper, a multi-device application is an application that employs multiple devices at the same time. Taking into account architecture of such applications, it is possible to distinguish two main flavors of multi-device applications: distributed and centralized multi-device applications. Of course, variations of such architectures are possible. The code of a distributed multi-device application is spread across multiple devices. These application fragments are executed at the same time and they collaborate to control the flow of the whole application. Such applications use solutions from research areas such as distributed computing and P2P communication, incorporating agents that provide both client and server functionality at the same time. In the case of a centralized multi-device application, an application is executed on a single device and uses services provided by other devices to carry out its functionality. This type of multi-device application uses classical client-server architecture and acts as a client to services provided by external devices that provide specific functionality (e.g., encryption), sensors, actuators, or human interface hardware such as a screen or a microphone.

Currently, a typical approach to multi-device application development is to prepare a vendor specific implementation that is locked into particular hardware or software platform. Examples of such multi-device applications include: Samsung SmartView (Samsung smartphone + Samsung TV), Automatic Link (iPhone + custom hardware attached to a car), mControl provided by Embedded Automation (mControl software on a mobile device + mControl home automation system), or Dice + (custom hardware dice + specially crafted game on a tablet or a smartphone). All these examples of multi-device applications are bound to specific devices or software platforms in the same way as mobile applications were a few years back. There are universal protocols, such as UPnP or ZeroConf, that allow connecting various devices, not only the ones provided by a vendor of a particular solution. But implementation of an application (or application fragments) is often still hardwired to specific hardware or software platform. The next step in the evolution of multi-device applications is to provide a unified execution platform or come up with tools and frameworks that will allow developing DIMDApps. So users would not be forced to buy a new set of specific devices, each time they decide to use a new multi-device application.

3 Definition of Key Requirements

Device independence and multi-device topics bring separate sets of requirements. The goal of device independence is to make functions of an application available on any suitable device without the need to modify the application itself. Following the Model-View-Controller pattern, commonly used to separate concerns of application implementation, it is possible to define three distinct requirements: device independence of application logic, device independence of application data, and device

independence of application user interface. On the other hand, the multi-device topic includes requirements tightly related to the architecture of a multi-device application. For distributed multi-device applications the most important requirement is communication between devices. For centralized multi-device applications it is access to various services and data provided by remote devices. A detailed analysis of each of these requirements is provided below.

3.1 Application Logic Device Independence

Application logic is a code that implements a behavior of an application (e.g., user authentication, data querying, calculations). To have a device-independent application logic means that a single implementation of an application can be used in the same way on any device. This is crucial if the application is supposed to be available to all potential users that use different types of devices with various operating systems. Commonly, the device independence of application logic is achieved by preparing cross-platform applications according to approaches described in Sect. 2.

3.2 Application Data Device Independence

Application data denotes files and other data processed by application logic or presented to a user. Application data device independence means that the data processing and presentation can be done regardless of a device used by a user. This aspect is important if a user often switches from one device to another and wants to maintain access the same application data. If the data (e.g., application configuration) is stored on a specific device then switching to another device makes the data unavailable. Or if the data requires special hardware or software to be processed or presented then its usage is limited to devices with such capabilities.

The device independence of application data can be achieved in one of several ways: by migrating the data from one device to another, by synchronizing the data across all user devices, or by providing a remote and persistent data repository that is reachable from any device. In all these cases it is important to encode the data in a common data format. The data format can be application specific, the only restriction is that the format should not depend on any device features. It is especially important for multimedia data such as video, due to specific codecs that might be required to present the data on a device. This issue can be addressed by using a standard data format that is widely supported (e.g., JPG for images), by agreeing on shared formats/codecs after handshake protocol, or by introducing an additional multimedia adaptation proxies such as the UMOD module developed for the ASIS system [17].

3.3 Application User Interface Device Independence

The term 'application user interface' covers all means of communication between an application and a user. The communication has two directions—from a device/application to a user (output) and back (input). Examples of output communication include a graphical user interface (GUI) presented on a device screen, with accompanying sounds and vibration effects. Input communication is related to user interface events caused by a user, for example: finger-based or stylus-based touch events and gestures, presses of physical buttons, or audio recording from a microphone. To have a device-independent application user interface (UI) means to have a single implementation of the application UI that can be used to communicate with a user regardless of a device used. This is a crucial aspect, because without a proper UI an application could be completely unusable even with fully device-independent application logic and application data.

The UI independence is a multi-layered problem analyzed by researchers working on UI adaptation problems [18]. The focus on the device independence aspect narrows the UI independence problem to creation of a platform-independent UI description and to its adaptation to capabilities of a particular device. The platform-independent UI description can be provided with standard and widely supported UI description languages such as XUL, OpenLaszlo, HTML/CSS or with abstract model-based UI description languages such as the one proposed by W3C Model-Based UI Working Group. In both cases the UI has to be adapted to a particular device. The XUL or HTML/CSS description can be adapted directly on the device with the use of JavaScript. An abstract model-based UI description requires an additional adaptation engine that will generate a final UI tailored to a specific device. Both solutions allow having a single UI description that can be used to consistently present an application UI on devices with different screen capabilities (size, resolution, form factor, etc.) or user interaction capabilities (finger-based or stylus-based touch interaction, special physical buttons, etc.). There is also a third approach, suitable for custom and complex user interfaces such as the ones used in games. In this case the UI is not described formally but it is generated on the fly directly by an application, which uses information about device capabilities to guide the UI generation process.

3.4 Communication Between Multiple Devices

Communication between multiple devices is important for distributed context-aware multi-device applications where the application logic is executed on different devices and it is necessary to exchange control messages and data between these devices. Without the communication it would not be possible to coordinate the execution of such a multi-device application and to gather context data from all possible sources. The communication can be implemented in one of two ways [19]: via a remote communication server (client-server architecture), or directly between

devices (peer-to-peer (P2P) architecture). The latter approach is more efficient and scalable, but requires additional infrastructure to provide information about other devices that should be taken into account in the communication process. It can be provided by a central register server, which makes it similar to the regular client-server approach, or by a P2P protocol, which is responsible for distributing necessary information in the network of connected devices.

As the communication between multiple devices is controlled by the application logic it is necessary to have the application installed and running on every communicating device. Alternatively, there must be a standard solution for launching an application to process a received message (such as, for example, the Intents feature provided by the Android mobile operating system).

3.5 Access to Features Provided by Devices

For centralized multi-device applications the important aspect is access to features provided by devices. In the centralized architecture there is only one instance of an application, running on a selected device or a server. There is no need to coordinate execution of multiple application fragments running on different devices, but it is important to have access to various features provided by devices other than the one running the application. These features include: output and input user interaction channels (e.g., a TV screen for displaying some data or a smartphone microphone for receiving a voice command), and various services that provide context data (e.g., remote geolocation service, battery status of a remote device, video stream from a camera of a remote device, remote encryption function that uses some specific hardware). Access to user interaction channels is necessary to interact with a user on multiple devices simultaneously and access to remote services is crucial when an application needs to use data that can be provided only by some remote devices—without the requirement to install the application on each of these devices.

This aspect of multi-device applications requires common communication protocols supported natively by devices or by execution platforms running on these devices. There are four protocols that have to be supported: 1/ output UI protocol for pushing UI fragments or UI updates to be presented on a particular device; 2/ input UI protocol for receiving notifications about UI events caught on remote devices; 3/ one-time service access protocol for incidental querying of remote services; 4/ continuous service access protocol for receiving notifications stream from a particular remote service.

4 Verification of DIA and Webinos Architectures

To verify the support for the DIMDApps requirements we have designed a sample application and we have implemented two variants of the application: one employing the DIA, which aims to facilitate achieving device independence, second

using the webinos platform, which provides universal execution platform and multi-device connectivity. Both implementations were then assessed in the context of DIMDApps requirements.

The application designed for this experiment is an electronic diary application called LifeLog. The LifeLog application allows recording digital memories in a form of short text messages, images, and audio recordings, along with context—user location and local time. It is possible to review all stored digital memories using the timeline view. The LifeLog application also facilitates sharing of recorded digital memories with other users. It can be done using the remote slideshow view, which allows presenting selected digital memories on screens of selected remote devices.

A device-independent application is supposed to be available on any suitable device, where suitable device is a device with capabilities required for intended user interactions. In the case of the LifeLog application it is a device capable of presenting a UI of the application (for the timeline and the remote slideshow views) and capable of receiving text, image and audio input from the user (for the timeline view). Therefore, it could be any device with a screen, speakers, a keyboard (either physical or virtual) or a microphone (plus speech-to-text functionality), a camera, and a microphone (for audio digital memories). In consequence, a suitable device for the LifeLog application is a PC, a smartphone, or a tablet. For the remote slideshow view the 'suitable device' definition includes also devices not intended for data input, such as a smart TV. Moreover, the LifeLog application includes both presented multi-device communication concepts and takes advantage of all available devices. It can use geolocation information provided by other user's devices (e.g. when adding digital memories on a PC the user location information can be provided by a smartphone) and it enables distribution of its UI to different devices (remote slideshow view). Moreover, the timeline view and the remote slideshow views of a particular user are synchronized on all used devices.

By designing the application in such a way we force the implementation to be device-independent and to use multi-device features. Therefore, this relatively simple application is sufficient for analyzing support for requirements of DIMDApps.

4.1 Device-Independent Architecture

The DIA, developed at the Department of Information Technology, Poznan University of Economics, is an approach to building applications available to users via any capable device from the large, diverse and fast growing pool of Internet-enabled end-devices—i.e., devices that are used directly by users to interact with an application. As presented by Chmielewski [3], the idea of Device-Independent Architecture originates from the Service-Oriented Architecture, where systems are decomposed into atomic services, and processes use these services without knowing much about their implementation. Similar approach can be used to

decompose end-devices. Each end-device, be it a laptop or a smartphone, provides: resources, services and user interaction channels. Resources encompass processing power, memory and storage. Services are providers of context information, such as location, temperature, light intensity, or data from other types of sensors. User interaction channels (input and output) include: screen, keyboard, vibration, camera, etc. The concept of DIA is based on an observation that a device acts as a proxy between an application and a user. Therefore, it is enough to provide a device with only two components: services and user interaction channels. The third component, resources used to execute an application, can be provided by a backed infrastructure.

Following this observation, the device independence provided by the DIA is achieved by separating an application from a particular end-device. The separation is done by: adopting the cloud computing paradigm and executing the application outside of the device, using only backend resources (ensuring device hardware and software independence), using abstract or universal user interface descriptions (ensuring device user interaction channel independence), and providing a middleware layer, which wraps all services with a standardized API and adapts application user interfaces to capabilities of available user interaction channels. The middleware functionality may be bound to the application itself as a software library, may be provided as a separate proxy service, or may be implemented in a form of a device independence driver (DID) directly on end-devices.

This approach to application-device separation resembles cross-platform approaches with their execution platform lock-in (and related fragmentation problems) and decreased performance due to additional abstraction layers. However, the DIA does not enforce usage of a particular application execution platform. Instead, DIA specifies a communication protocol and allows application developers to choose any backend execution environment for their applications. In terms of performance, the use of DIA introduces additional delay, but it's characteristics is different than in cross-platform approaches. In cross-platform approaches the goal is to separate application logic from the hardware—therefore slowing down the application logic execution. In DIA the separation is between the application logic and the user interface presented on an end device, so it is only a UI response time that may suffer, not the actual performance of application logic execution. Early research in this area [20] shows that the UI response time of DIA-based applications can be maintained under 100 ms (or under 1 s on slower GSM-based communication channels), which is almost unnoticeable for users [21].

Moreover, thanks to the presented device decomposition, the DIA provides an additional feature important for DIMDApps. Applications based on the DIA operate on sets of services and user interaction channels instead of specific devices. In consequence, a single application may use services and user interaction channels provided by multiple devices—enabling implementation of various multi-device use cases [22].

The implementation of a DIA-based LifeLog application uses the fact that target devices for the LifeLog application are smartphones and tablets. It is safe to assume that all such devices are equipped with a Web browser, which can be used as a DID

and a cross-platform UI handler. Additionally, despite the common Web platform providing universal UI format and UI optimization capabilities through JavaScript, server-side UI adaptation has been introduced to improve the usability of the application on different devices. The DIA enables multi-device usage scenarios, but does not provide any direct support for implementing such scenarios. Therefore, intra-device synchronization functions required by the LifeLog application had to be manually implemented using WebSockets.

Elements of the application presented in Fig. 1 are the following:

- DEV T—end-device of user A with a Web browser displaying the timeline view,
- DEV S1—end-device of user A with a Web browser displaying the remote slideshow view,
- DEV S2—end-device of user B with a Web browser displaying the remote slideshow view,
- SERVER—Web application server, data storage, and WebSocket communication server.

End-devices (DEV) are responsible only for presenting the application UI and handling UI interactions. The application UI is generated on the server, described with Web technologies (HTML5, CSS3, and JavaScript), and sent to end-devices using HTTP. All data are stored on the server, which also acts as a communication proxy distributing view synchronization messages (via a WebSocket connection).

Application logic device independence Since in DIA the application is executed on a backed infrastructure and not on end-devices, the application logic is fully device-independent. Moreover, developers are free to choose any backend application execution environment. In the DIA-based LifeLog application the application logic is implemented in PHP and executed by a PHP module of an Apache Web server running on a Linux-based virtual machine.

One can argue, that devices still need to provide a uniform platform for accessing user interaction channels and services. However, the DIA requires only that a device exposes its services and user interaction channels according to predefined protocols. This can be provided by a native implementation of these protocols or the DID, which is independent of the logic of an application and is simple

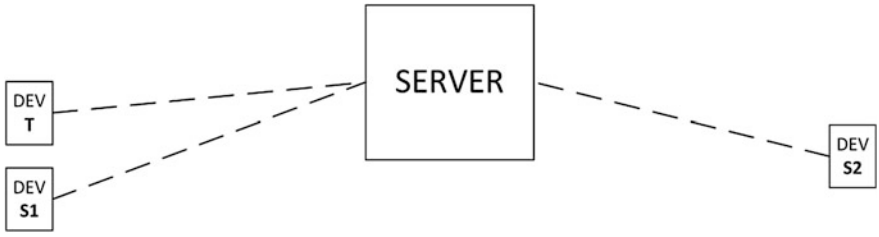

Fig. 1 Architecture of the DIA-based LifeLog application

enough to be easily ported to any device and software platform. The DIA-based LifeLog application uses Web browsers as a target platform to facilitate implementation of the DID. However, even with such a common platform, different Web browsers require slightly different implementations of the DID. Notably in the field of media capture, because various Web browsers differ in the implementation of the related W3C API. Nevertheless, the DID is not part of the application and its implementation does not influence the device independence of the LifeLog application logic.

Application data device independence The DIA assumes that all application data and files are stored using a backend storage—independently of devices used by an application. The DIA-based LifeLog application uses a MySQL database and a file repository to store digital memories. Both these data storage components are located on a server.

The DIA does not favor any particular data formats, so it is up to the developer to choose data formats that will ensure application data device independence. The DIA-based LifeLog application uses Web browsers for UI and data presentation, so the set of device-independent data formats equals standard Web formats. In this case it is JPEG for images and WAV for audio (converted on the server from a device-specific recording format (e.g. 3GPP) provided by a device).

Application user interface device independence Although, the DIA includes a concept of a UI adaptation layer, it does not force developers to use a particular UI description language or UI adaptation strategy. In the simplest case DIA-based application may use a universal UI description language and no UI adaptation. More complex UI solutions depend on implementation of a middleware responsible for UI adaptation [23]. The DIA-based LifeLog application uses HTML-based UI descriptions generated on the server-side, plus simple adaptation done directly on the client-side. Since, the UI described this way will be presented properly on any target device, it can be stated that the DIA-based LifeLog application has a device-independent UI.

Communication between multiple devices In DIA an application is running in the cloud, not on end-devices, so there is no need to pass messages and data directly between end-devices. The only application component distributed to end-devices is the application UI. In multi-device scenarios UI fragments presented on different devices might need to be synchronized. The DIA assumes that such synchronization is controlled in a centralized manner by the application. In the DIA-based LifeLog application, the UI synchronization is provided by a WebSocket server. It is a separate component of the LifeLog application logic, but this separation is caused by technical characteristics of the targeted platform, not the DIA.

Access to features provided by devices The DIA supports access to features provided by devices only to the extent of services supported by the DID. This limits the ability of a multi-device application to use sources of data not based on end-device sensors (e.g. encryption, data conversion, etc.). In case of such sources

the application must implement the access functionality on its own. The DIA-based LifeLog application uses only external geolocation service, which is supported by the DID.

4.2 Webinos Platform

The webinos platform is an open source Web application execution environment that enables applications to be run securely on and across multiple connected devices. A strong consortium of about 30 members coordinated by Fraunhofer FOKUS supports this key delivery within the EU-funded research project named "webinos". The fundamental goal is to make services and resources to be used and shared consistently and securely over a broad spectrum of connected device domains covering TV, desktop, vehicle, mobile, and Internet of Things. As a multi-device middleware webinos fosters compatibility and interoperability for applications by establishing a user centric overlay network. This network consists of so called interconnected Personal Zones (PZ). A PZ is the most significant concept in webinos defining a set of disparate devices and services owned by a particular user. The key feature of the PZ allows applications and devices to securely discover and make use of each other's services and resources while preserving the user's full control regarding the access to functionality and data. This access control is carried out by a fine-grained policy system. An application developed for the webinos platform is an installable packaged Web page [24] also known as a widget written using Web technologies such as HTML, CSS, and JavaScript. Using specified APIs provided by webinos [25] the actual location of a service is transparent for the developer, e.g., there is no difference if a file is stored on a remote device or on the same device where the application is executed when accessed via the webinos File API. The execution environment for webinos applications is the Web runtime (WRT) that connects to the Personal Zone Proxy (PZP). The PZP acts as an agent and is a mandatory component on every webinos enabled device. This agent enables, among others, messaging, local and remote service access, and policy enforcement. The communication between mutually authenticated personal devices may be routed through one or more Personal Zone Hubs (PZHs) or may be direct between two peers if authorized by prior certificate exchange. The PZH being instantiated once per user is responsible for the synchronization and message routing within the user's PZ and inter-zone service discovery and communication if several PZHs are interconnected.

The abstraction from service location results in an independence on local device capabilities enabling cross-platform development. Instead of limiting the capability set for applications to a common minimum that is available on all involved devices, the remote utilization of capabilities is encouraged. The development model is based on a single virtual device that comes with consolidated properties (e.g., service or resource) of all connected and accessible devices while the availability is determined dynamically at runtime.

The multi-device webinos platform supports applications installed locally on a device and implemented with Web technologies. The WRT used to execute applications is essentially a Web rendering engine, similar as in Web browsers, providing access via the PZP to device features and inter-device communication, in a distributed setting through local bearer technologies. Consequently, on this platform, the LifeLog application has been implemented as a local Web application—without the server layer. The multi-device features of the LifeLog application have been implemented using APIs provided by the webinos platform.

Since the webinos platform is based on Web technologies it inherits all device independence benefits introduced by the common Web standards. The application logic implemented in JavaScript can be executed on any device equipped with the WRT. The user interface is universally described with HTML and CSS, as in the DIA-based LifeLog variant. There is no server layer, so it is not possible to perform a server-side adaptation of the LifeLog application UI, but in this case a consistent UI across different devices is provided by UI optimization done with JavaScript.

One issue that required a custom solution is currently data storage. At this moment, the webinos File API provides a file storage remotely accessible but bound to a particular device. To maintain proper device independence of the LifeLog application it has to be assumed that there is an additional persistent device holding files stored using the File API. An interesting side effect of such a design is that a user stays in full control of her data.

Elements of the application presented in Fig. 2 are the following:

- DEV SS—webinos-enabled persistent device providing the data storage for user's digital memories,
- DEV T—webinos-enabled end-device of user A with the LifeLog application displaying the timeline view,
- DEV S1—webinos-enabled end-device of user A with the LifeLog application displaying the remote slideshow view,
- DEV GL—webinos-enabled end-device of user A providing additional geolocation service,

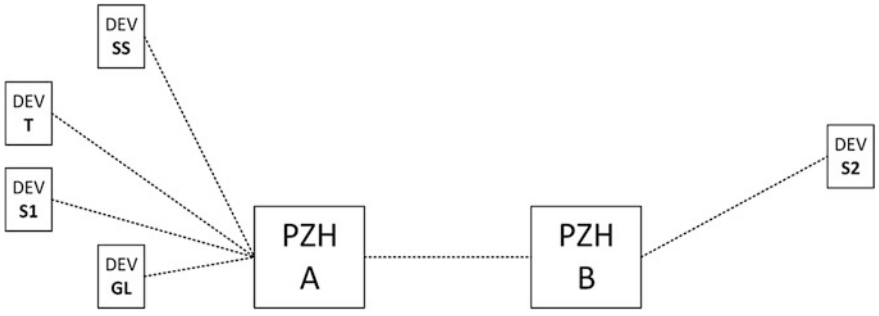

Fig. 2 Architecture of the webinos-based LifeLog application

- DEV S2—webinos-enabled end-device of user B with the LifeLog application displaying the remote slideshow view,
- PZH A—remote, persistent Personal Zone Hub of user A,
- PZH B—remote, persistent Personal Zone Hub of user B.

All devices are webinos-enabled, which means that they run a PZP and use WRT to execute the LifeLog application. All processing and UI adaptation is done locally on each device. Application data are stored on the DEV SS within the PZ of user A. Devices communicate with the PZH using a bidirectional TLS connection, whereas the communication between the WRT and PZP on each device is handled via WebSockets.

Application logic device independence The webinos platform addresses the application logic device independence requirement by delivering a universal execution environment based on Web standards that is not limited to a subset of services provided by a single device. The WRT provides a set of APIs for universal access to functions and data provided by a local device and remote devices. The webinos approach to device independence is the same as existing cross-platform approaches, with performance issues and usage limitations. The application logic execution performance might be hindered on some low-end devices, due to the additional abstraction layer, but this effect was not noticeable for the LifeLog application. Usage limitations include still limited platform availability and a single supported programming language—JavaScript. Current reach of the WRT is limited to eight OS platforms of five different device domains. However, the portability to further platforms is facilitated by the publicly available open source assets for all relevant webinos platform components.

The webinos-based LifeLog application logic is implemented with JavaScript and can be executed by the WRT installed on any webinos-enabled device. In consequence, the webinos-based LifeLog application logic device independence is limited by the reach of the webinos platform.

Application data device independence At the moment, the webinos platform does not provide a way to store data independently of a device. Each webinos-based application is installed on a particular device and stores its data on one of user's devices. In consequence, when the device storing the data is not available, other devices are not able to access the data. However, if a user's PZ includes at least one persistent device (e.g., webinos-enabled NAS) then this device could be used to store data and provide a base for application data device independence. The webinos-based LifeLog application employs this approach and uses a dedicated, persistent device to store user's data.

The webinos platform, similarly to the DIA, does not favor particular data formats, so it is up to the developer to choose data formats that will ensure application data device independence. The webinos platform is based on Web standards, so the set of device-independent data formats by the webinos-based LifeLog application is the same as in the DIA-based LifeLog variant.

Application user interface device independence The webinos platform is based on Web technologies. Therefore, webinos-based applications may use HTML and CSS standards to provide universal and device-independent UI descriptions, which can be further optimized to capabilities of a particular device with CSS Media Queries and custom JavaScript functions executed on an end-device. The webinos-based LifeLog application uses HTML-based UI descriptions, plus simple UI optimization done with CSS Media Queries. Since, the UI described this way will be presented properly on any webinos-enabled device, it can be stated that the webinos-based LifeLog application has a device-independent UI.

Communication between multiple devices On the webinos platform the communication among devices is based on the P2P architecture. A PZH is used as a synchronization and message routing server, but when the App2App Messaging API is used the PZH is transparent to the application. In cases where Internet connectivity is not available the devices may opt in into a direct P2P communication, PZP to PZP, while the usage of the APIs remains the same as if they were connected via the PZH. The webinos-based LifeLog application uses the application level communication (App2App Messaging API) for synchronizing views and controlling remote slideshow views.

Access to features provided by devices The webinos platform supports service access protocols with the Discovery API, which allows an application to find and use all instances of a requested service irrespective of a device providing it. A service may be provided by a local or by a remote device. It is transparent to the application and every instance of a service can be called in the same way. If a service is not directly available it is possible to launch an application using the AppLauncher API and use application level communication. The webinos-based LifeLog application uses the Discovery API to find available geolocation services and queries them using Geolocation API. It is not possible to display the remote slideshow view without the application instance running on a device, but there are plans to cover at least part of this functionality with the future RemoteUI API. Also, because of the Web nature of webinos, it may benefit from new Web-based solutions such as the Presentation API proposed by the W3C Second Screen Presentation Community Group.

5 Definition of a Combined Architecture

The presented architectures approach existing problems differently and they do help building and running DIMDApps. But both have their drawbacks. The DIA approach to device independence based on common protocols, instead of a common execution engine, avoids the reach and fragmentation problems of today's cross-platform solutions. The backed data storage ensures device independence of application data. And the separation of UI generation and presentation makes it possible to employ various UI adaptation strategies. However, the DIA does not

provide much support for multi-device applications. Synchronization of distributed UI fragments has to be done manually and support for access to features provided by devices is limited to predefined services based on end-device sensors (data providers). Access to features that offer data processing (e.g. hardware-based encryption, multimedia transcoding) has to be implemented manually. On the other side, communication solutions provided by the webinos platform help implementing multi-device applications. But the webinos architecture follows the traditional approach to device independence (universal execution platform), with all its drawbacks. The common execution engine (WRT) is based on universal Web technologies, but it has to deal with the reach and fragmentation problems and it limits developers to a single programming language. Moreover, the focus on end-devices makes it impossible to implement a truly device independent application data layer, as the webinos architecture does not include any backed infrastructure that could act as a persistent and device-independent data storage. A summary of DIA and webinos support for DIMDApps requirements is presented in Table 1. The support has been graded using a 3-points scale: 3 points for full support, 2 points for support with drawbacks, 1 point for almost no support, 0 points for no support at all. Each point is represented by a + sign.

The summary makes it evident that DIA and webinos are supporting different subsets of DIMDApps requirements. Combination these two approaches should provide a solid platform for DIMDApps. Therefore, we propose a Device-Independent Webinos Architecture (DIWA) presented in Fig. 3. In this architecture

Table 1 Summary of DIA and webinos support for DIMDApps requirements

	DIA	webinos
Application logic device independence	+++	++
Application data device independence	+++	
Application UI device independence	+++	++
Communication between multiple devices		+++
Access to features provided by devices	+	+++

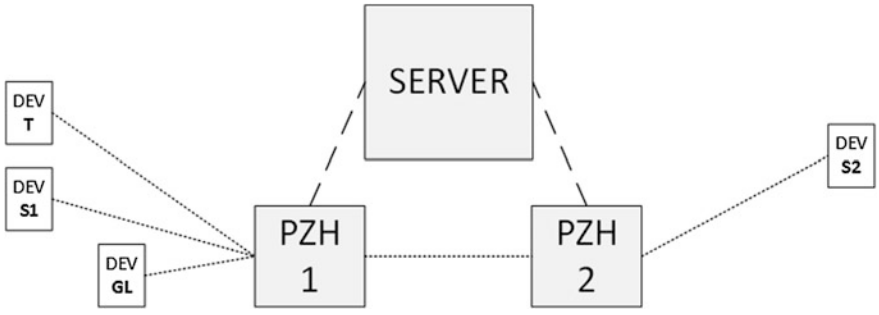

Fig. 3 Architecture of the DIWA-based LifeLog application

device-independence requirements are satisfied using solutions provided in the DIA and support for multi-device requirements is based on the webinos.

Application logic is executed on the SERVER, which communicates with user's PZHs using universal protocols. This way, developers are free to use any application execution platform without constraints on a programming language or an application runtime. This approach also avoids the reach and fragmentation problems.

Application data is stored on the SERVER or is using any other persistent data repository solution (e.g. external services such as Dropbox). A set of available data formats is not limited and it is up to the developer to choose data formats that will ensure application data device independence.

Application UI generation is also done on the SERVER. The application UI is then transferred (via PZHs) to DEVs for presentation. The transfer is done using protocols specific to user interaction channels used by an application. Optional UI adaptation can be done between the SERVER and a PZH. Additionally, PZHs help distributing and synchronizing UI fragments presented on different DEVs. UI presentation and access to features provided by various DEVs is supported by a modified webinos runtime, which acts as the DID.

The DIWA provides the best elements of both analyzed solutions. Therefore, the LifeLog application implementation on this architecture could easily provide all its functions. The server part includes application logic and data, to ensure high level of device independence. The client part is based on the modified WRT (DID) and with the help of PZH supports multi-device usage scenarios.

Elements of the architecture presented in Fig. 3 are the following:

- DEV T—end-device of user A with the LifeLog application displaying the timeline view,
- DEV S1 (DEV S2)—end-device of user A (B) with the LifeLog application displaying the remote slideshow view,
- DEV GL—end-device of user A providing additional geolocation service,
- SERVER—remote application server and data storage that can be implemented with any persistent backend infrastructure (e.g. cloud-based or a specialized device),
- PZH A, PZH B—remote, persistent PZHs of user A and B.

In the DIWA the WRT does not have to be a full featured runtime anymore—just a DID. DID internal logic is limited to managing access to user interaction channels and services. So the sophisticated WRT, previously responsible for executing applications and rendering HTML-based UIs, becomes quite simple to implement. This simplification helps porting the WRT to new end-devices—also low-end embedded devices that are a hard target for the webinos platform.

6 Summary and Future Work

In this paper we have targeted a problem of developing applications that may employ multiple end-devices—possibly any end-devices. We have proposed a set of five requirements that have to be satisfied to build a truly context-aware device-independent multi-device application (DIMDApp):

- application logic device independence,
- application data device independence,
- application user interface device independence,
- communication between multiple devices,
- access to features provided by devices.

Following these requirements we have examined two distinct application architectures that are supposed to facilitate implementation of DIMDApps: the Device-Independent Architecture (DIA) and the webinos architecture. Both solutions have been analyzed on a base of a context-aware device-independent multi-device LifeLog application. This case study highlighted pros and cons of both architectures in terms of their support for DIMDApps requirements. A comparison of the analyzed architectures revealed that their support for DIMDApps requirements is complementary and that it is possible to formulate a combined architecture that satisfies all requirements. The result is a definition of Device-Independent Webinos Architecture (DIWA), which takes best solutions from DIA and webinos, and delivers a solid foundation for building truly device-independent multi-device applications.

Although, the proposed DIWA satisfies all stated DIMDApps requirements, it does not solve all the problems related to DIMDApps implementation. There are at least three aspects that require further research: performance, reach, and UI adaptation. The DIWA uses the device-application separation approach proposed by DIA, which changes the application logic execution performance problem into the application UI response time problem. As it was already stated, the initial research shows that it is possible to maintain reasonable response time, but it is necessary to find out what are the key constraints influencing the UI response time in this architecture and to learn how to minimize or avoid them. In DIWA, the reach problem is addressed by the simplification of the WRT. Simple native component is easier to implement and to port to new platforms, but it would be beneficial to simplify it even more or to define universal protocols instead of the implementation. Key role of this native component is to manage access to UI channels and services provided by an end-device (e.g. context data sources). By researching generic protocols for accessing these channels and services it should be possible to specify only these protocols—not how they must be implemented. The last bit is UI adaptation. Although DIWA enables the use of UI adaptation mechanisms, they are not included in the solution. It should be explored what UI adaptation approaches are optimal for different types of applications and how to embrace them in the DIWA.

References

1. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, Cisco (2014)
2. The New Multi-screen World: Understanding Cross-platform Consumer Behavior, Google Think Insights blog. http://ssl.gstatic.com/think/docs/the-new-multi-screen-world-study_research-studies.pdf, Google (2012)
3. Chmielewski, J.: Towards an architecture for future internet applications. In: The Future Internet, pp. 214–219. Springer, Berlin (2013)
4. Lyle, J., Faily, S., Flechais, I., Paul, A., Goker, A., Myrhaug, H., Desruelle, H., Martin, A.: On the design and development of webinos: a distributed mobile application middleware. In: Proceedings of the 12th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2012), pp. 140–147. Springer, Berlin (2012)
5. Puder, A., Römer, K., Pilhofer, F.: Distributed systems architecture: a middleware approach. Morgan Kaufmann Publishers, San Francisco (2006)
6. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., Nahrstedt, K.: Gaia: a middleware platform for active spaces. ACM SIGMOBILE Mob. Comput. Commun. Rev. **6**(4), 65–67 (2002)
7. Ponnekanti, S.R., Johanson, B., Kiciman, E., Fox, A.: Portability, extensibility and robustness in iROS. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, pp. 11–19. IEEE (2003)
8. Seffah, A., Javahery, H. (eds.): Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. Wiley, New York (2005)
9. Android Developer Best Practices: Supporting Multiple Screens, Android Developers website. http://developer.android.com/guide/practices/screens_support.html, Google (2015)
10. Mobile Platforms: The Clash of Ecosystems, VisionMobile research blog. http://www.visionmobile.com/product/clash-of-ecosystems/, VisionMobile (2011)
11. Android Developer Dashboards, Android Developers website. http://developer.android.com/about/dashboards/index.html, Google (2015)
12. Bacic, M., Chmielewski, J.: Usefulness of New HTML5 and CSS3 Features for Internet Applications on Mobile Devices. In: Information Technologies in Organizations—Management and Applications of Multimedia, pp. 97–110. Wydawnictwa Towarzystwa Naukowego Organizacji i Kierownictwa - Dom Organizatora, Toruń (2013)
13. Cross-Platform Tool Benchmarking 2014, research2guidance website, http://research2guidance.com/cross-platform-tool-benchmarking-2014/, research2guidance (2014)
14. Cusumano, M. A., Yoffie, D. B.: What netscape learned from cross-platform software development. Commun. ACM **42**(10), 72–78 (1999)
15. Cross-Platform Developer Tools report, VisionMobile research blog. http://www.visionmobile.com/product/cross-platform-developer-tools-2012/, VisionMobile (2012)
16. Ohrt, J., Turau, V.: Cross-platform development tools for smartphone applications. Computer **45**(9), 72–79. IEEE (2012)
17. Walczak, K., Wiza, W., Chmielewski, J.: Adaptation of user interfaces in SOA applications. e-Minds: Int. J. Hum. Comput. Inter. **2**(8), 3–17 (2012)
18. Calvary, G., Coutaz, J., Thevenin, D., Bouillon, L., Florins, M., Limbourg, Q., Souchon N., Vanderdonckt, J., Marucci, L., Paternò, F., Santoro, C.: The CAMELEON reference framework, Cameleon project. http://giove.isti.cnr.it/projects/cameleon/pdf/CAMELEON D1.1RefFramework.pdf (2002)
19. Tanenbaum, A., Van Steen, M.: Distributed Systems. Pearson Prentice Hall, Upper Saddle River (2007)
20. Chmielewski, J.: Device-independent architecture for ubiquitous applications. Pers. Ubiquit. Comput. **18**(2), 481–488 (2014)

21. Nielsen, J.: Response times: the 3 important limits, Jakob Nielsen's Alertbox. http://www.nngroup.com/articles/response-times-3-important-limits/ (1993)
22. Chmielewski, J., Walczak, K.: Application architectures for smart multi-device applications. In: Proceedings of the Workshop on Multi-device App Middleware, pp. 1–5. ACM, New York (2012)
23. Jansen, A., Bronmark, J., Chmielewski, J.: Method of adapting a user interface in industrial process monitoring and control applications, The Swedish Patent and Registration Office, SE 1300702–6 (2013)
24. Packaged Web Apps (Widgets)—Packaging and XML Configuration, 2nd edn. http://www.w3.org/TR/widgets/, W3C (2012)
25. webinos Device APIs, http://dev.webinos.org/specifications/api/, webinos (2013)

Performance Analysis of Web Systems Based on XMLHttpRequest, Server-Sent Events and WebSocket

Wojciech Słodziak and Ziemowit Nowak

Abstract The aim of the authors of this chapter is to analyze the performance of Web-based systems using XMLHttpRequest, Server-Sent Events and WebSocket. Dedicated tool was presented enabling the performance of research for the above technologies. The diagrams illustrating the method of operation of the tool in the event of the investigation of transmission from the client to the server, from the server to the client and in both directions were presented. The results of the research conducted with the help of dedicated tool in the local network and on the Internet for two web servers and four browsers. The results obtained were discussed and appropriate conclusions drawn. The directions for further research were pinpointed.

Keywords Web systems · Performance analysis · XMLHttpRequest · Server-sent events · Websocket

1 Introduction

The performance of web-based systems may be improved in a number of ways, one of which may be through the use of the existing solutions that differ in terms of application use. In the article [1] the authors address the issue of improving the operation of applications supporting remote collaboration through practices used in creating online games. Problems such as packet loss, poor coverage in the case of mobile and wireless networks, and limited transmission bandwidth are solved by the creators of multiplayer games and their experience can be used e.g. in order to build systems supporting remote collaboration.

Many of the problems associated with the reduced network speed and large size of transmitted messages can be solved by means of data compression. The article [2] describes, and includes studies in GMC (General Message Compressor) compression technology. The developers of the technology prove the effectiveness of

W. Słodziak · Z. Nowak (✉)
Wrocław University of Technology, Wrocław, Poland
e-mail: ziemowit.nowak@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_6

their creation through the research contained in the article. The use of the tool enables the reduction in the size of the common message formats up to 20 % of the original size for messages based on simple text, and up to 8 % for XML format. Compression is one of the many areas of the complex process of data transmission, in which improvements can be looked for.

The selection of appropriate data transmission technologies tailored to the needs of the application is very important for developers of Web applications. This issue constitutes the main theme of this chapter and is widely discussed in literature [3, 4, 5]. In article [5] the authors examine the performance of applications built on the basis of the following technologies: XHR, Java Applet and WebSocket. In the tests the number of messages sent per second in the direction from the browser to the server and the other way round was chosen as a performance criterion. According to the research, the authors designated WebSocket as the technology that yields the best results.

In view of the above facts, the study and analysis of the available technologies and tools enable the creation of better and better Web application solutions and the movement away from restrictive desktop applications in favour of efficient and readily available web applications. The performance of Web applications is a multifaceted problem and so the improvements are to be searched for in many areas.

The aim of the authors of this chapter is to analyze the performance of Web-based systems using XMLHttpRequest [6], Server-Sent Events [7] and WebSocket [8, 9]. Another issue raised is the characteristics of strengths and weaknesses of the technology and identification of applications in which the technology performs at its best.

2 The Description of Performance Measurement Tool

For the purposes of the tests dedicated tool was developed to measure the mean time of sending a message or a pair of messages, depending on the technology used, and the testing scenario. The tool consists of the client (on the side of the browser) and the server parts.

The client part is used to generate messages and to control the tests and display the results. To this end, the user interface was created which executes actions on the side of the client and the server. Through the interface, test sequences may be run, it is also possible to read the results and control the parameters that influence the number and type of messages sent to and from the server. The parameters that can be set for each of the investigated technologies include:

- the number of messages sent in a sequence,
- the type of message:
  - fixed size message—the size in bytes to be specified,
  - random size message—the range of minimum and maximum in bytes to be specified,

– growing size message—growth in bytes of each subsequent message to be specified.

The server part is responsible for receiving and sending messages to the browser. In order to transmit messages from the server to the client, the methods of generating messages in the same manner as in the case of the client were implemented.

Tool was designed in such a way as to enable the practical comparison of the technologies surveyed. To this end three modules were implemented to compare the various combinations of the technologies with the same application:

- communication from the client to the server,
- communication from the server to the client,
- bidirectional communication.

In each module two sets of technologies were used meeting the same requirements, but differing in the implementation itself.

2.1 Communication from the Client to the Server

For communication from the client to the server XMLHttpRequest and WebSocket technology was used (Fig. 1). WebSocket also allows sending messages in the opposite direction, which was not used in this combination.

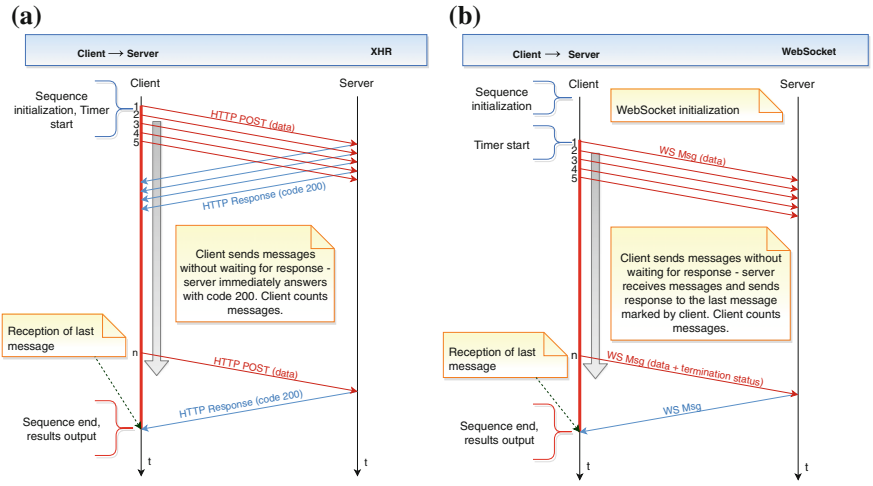

Fig. 1 The diagram of communication from the client to the server: **a** XMLHttpRequest technology, **b** WebSocket technology

In the case of XMLHttpRequest technology it is not necessary to establish a special connection between the browser and the server beforehand, as receiving HTTP protocol messages is a standard Web server functionality.

The situation is different for WebSocket technology, which requires the initialization on the part of the client (the browser). The browser reports its willingness to open TCP connections (via JavaScript), to which the server consents, provided the application, which is running on it, has such functionality implemented. The connection thus formed is maintained throughout the communication.

2.2 *Communication from the Server to the Client*

Server-Sent Events and WebSocket technologies were used for communication from the server to the client (Fig. 2). Both the technologies allow sending messages from the server to the browser (client). WebSocket also allows sending messages in the opposite direction, which was not used in this combination.

Server-Sent Events technology is based on HTTP protocol. The browser initiates the connection by creating (by JavaScript) an EventSource object listing in the URL parameter, to which a Java servlet implementing the SSE is assigned. The server, in order to establish a Keep-Alive connection, returns a partial HTTP response with HTTP Content-Type = text/event-stream header.

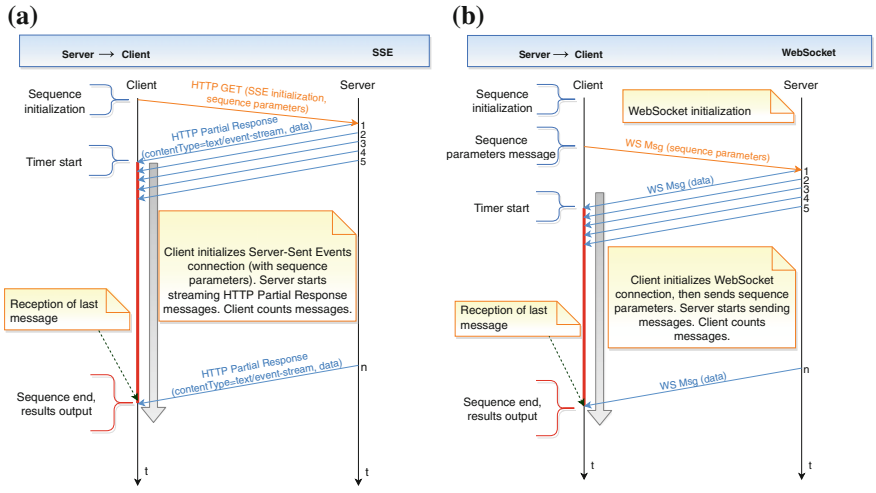

Fig. 2 The diagram of communication from the server to the client: **a** Server-sent events technology, **b** WebSocket technology

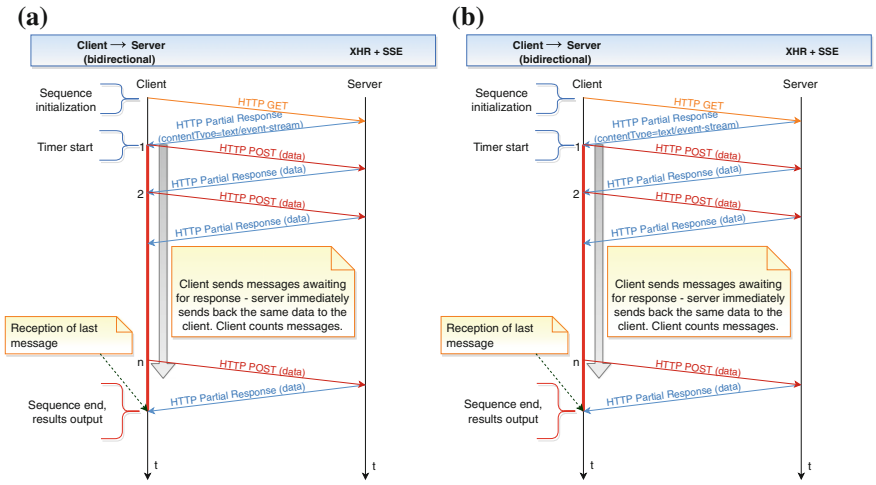

Fig. 3 The diagram of bi-directional communication: **a** Server-sent events + XMLHttpRequest technologies, **b** WebSocket technology

2.3 Bidirectional Communication

Server-Sent Events in conjunction with the XMLHttpRequest and WebSocket technology was used for the purposes of bidirectional communication (Fig. 3). The first set of technologies enables to sequentially send messages from the server to the client and from the client to the server, which, in combination, makes bidirectional communication possible. WebSocket is a technology, which itself ensures such communication.

The combination of Server-Sent Events and XMLHttpRequest technologies is a solution fully based on the HTTP protocol. The initialization of such communication requires, as before, that JavaScript and a special Java servlet that implements SSE be used. To transmit messages from the client to the server does not require an additional implementation except retrieving HTTP Post messages.

3 Performance Tests

The study involved two servers, Apache Tomcat and Glassfish which have an in-built module for enabling the Web container to run Java servlets. The servers differ from each other in that Tomcat is merely a container for servlets and Glassfish is also a container for Java EE, which extends its functionality. It was possible to use other, more popular servers, but it would have required a Web-based container

module to be installed. The study was carried out using two computers, one of which had application servers installed and running for testing, and the other the current versions of the four popular web browsers:

- Chrome—version 43,
- Firefox—version 37,
- Opera—version 29,
- Internet Explorer—version 11.

The computers used were characterized by the following parameters:

- The Server
  - Operating System—Windows 7 Pro x64,
  - Processor—Intel Core 2 Duo T6600,
  - Memory—4 GB RAM,
  - Hard drive—300 GB.
- The Client
  - Operating System—Windows 8.1 x64,
  - Processor—Intel Core i7-4710HQ 2.5 GHz,
  - Memory—8 GB RAM,
  - Hard drive—128 GB SSD.

The studies were conducted using a LAN and a WAN network. During the test involving the LAN network, the connectivity between the client and the server was provided by a WiFi access point, functioning simultaneously as a router. During the test involving a WAN, the client was connected to the Internet through a WiFi access point of the Eduroam network of the Wrocław University of Technology, and the server via a WiFi access point of the Neostrada network (ADSL).

Before starting the target tests, the initial simulation was conducted to determine the number of messages sent in the test sequence, which would give satisfactory results. To do so, the client-to-server module and XHR technology a number of tests were carried out with a variable number of messages sent in a test sequence. The selected amount was 1000, due to the small difference in the standard deviation from the number of 2000, and due to the need to shorten the test time, directly proportional to the number of messages per sequence. It should be noted that the greater the number of messages in a testing sequence the better the test results as the error associated with the current noise arising due to additional processes performed on both machines and accidental network load is reduced.

The target research was carried out for the three communication modules implemented in tool.

The result of the test sequence was the average time from the transmission of a thousand of individual or pairs of messages (in the case of bilateral communication). The time was measured, as shown in the drawings describing the

communication within the tool (Figs. 1, 2 and 3). The research conducted involved the following parameters that were adjusted:

- client (browser),
- server,
- type of network (LAN/WAN)
- number of messages in a sequence.

Due to the extent of the studies only selected results will be presented.

3.1 Communication from the Client to the Server

The first studied type of message transmission was the communication from the client to the server. In this case tool lets one use XMLHttpRequest and WebSocket technologies.

In Fig. 4 you can see a general tendency spread of results between the local and the global network. It is logical that the selected technology achieves better results for the local network than for the global one, as in the local network communication is almost instantaneous, due to the short way there is for packages to travel. In addition, the communication is accompanied by lower noise associated with the network load, which generates more stable results.

For Tomcat server XMLHttpRequest technology obtains worse results than WebSocket for both the local and global networks.

Figure 5 shows the comparison of the performance of a given technology on Tomcat and Glassfish servers. it shows that when it comes to the Tomcat server XMLHttpRequest technology is always slower than in the case of WebSocket. When it comes to the Glassfish server both the technologies have similar times and

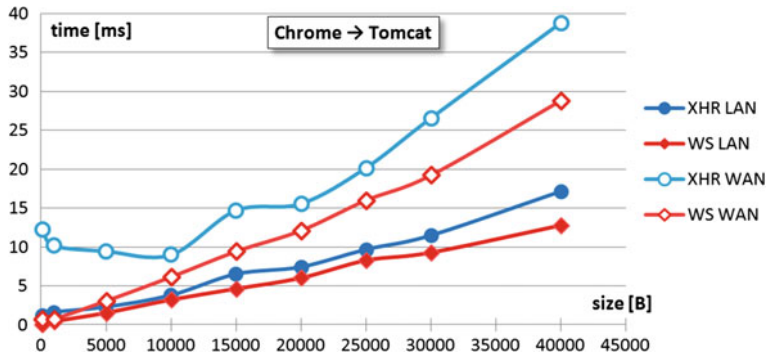

Fig. 4 The time of message transmission depending on its size for Chrome browser and Tomcat server

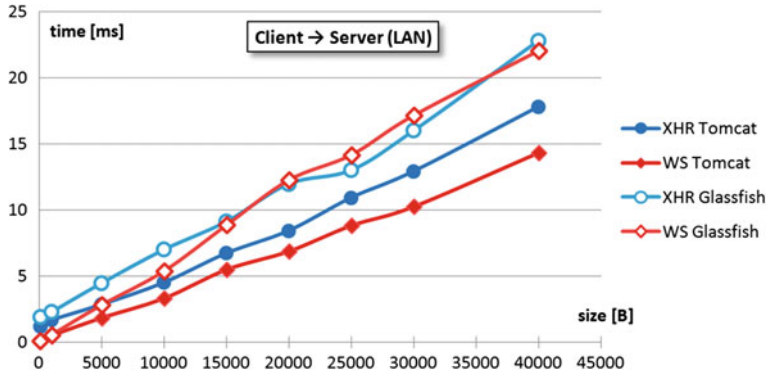

Fig. 5 The time of message transmission on the local network depending on message size (arithmetic average of the performance of individual web browsers)

intertwine in the two places of the graph. WebSocket has only a big advantage for small sized messages.

From the diagram it can be concluded that the Tomcat server is more effective when it comes to the communication from the client to the server. In any case, the same technology copes better on the Tomcat server. What is more, for the majority of the researched sizes of messages XHR technology on the Tomcat achieves shorter transmission times than WebSocket on the Glassfish server. The opposite result was obtained only in the case of message sizes smaller than 5000 bytes.

3.2 Communication from the Server to the Client

The communication module from the server to the client of tool is used to compare Server-Sent Events and WebSocket technologies. As Internet Explorer does not support SSE technology, the former was excluded from further study.

Figure 6 shows the average time for the transmission of messages from the Tomcat server to Opera browser depending on the size of the message. The careful observer will notice that WebSocket technology achieves better results for larger sizes of messages for the global network, than for the local network. This is an unexpected result, since LAN is not accompanied by a delay in contrast to the global network. This result may have been caused by random phenomena, such as e.g. accidental load on the network or CPU. It does not occur with every browser tested.

Another observation, which applies also to the other browsers, is a significant increase in the average transmission time of sending messages for Server-Sent Events technology over WAN for the message sizes larger than 7500 bytes. This phenomenon was not observed in the tests performed using the Glassfish server, as can be seen in Fig. 7.

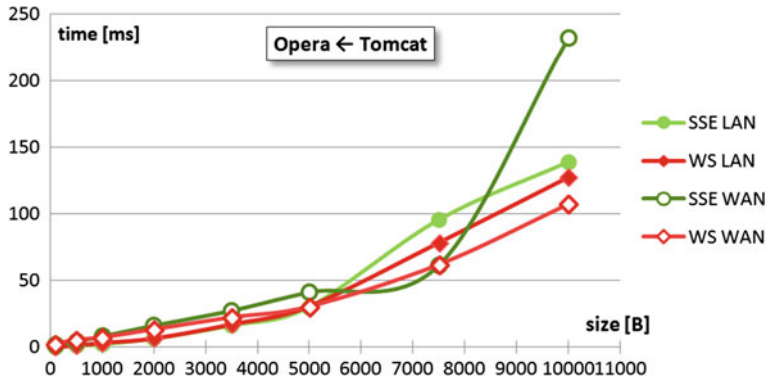

Fig. 6 The time of message transmission depending on its size for Opera browser and the Tomcat server

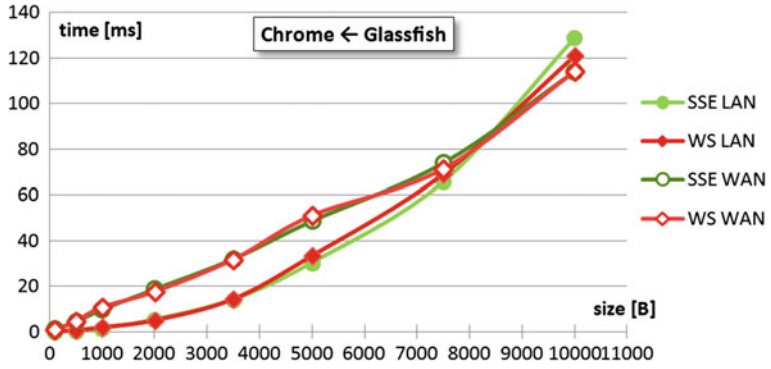

Fig. 7 The time of message transmission depending on its size for Chrome browser and the Glassfish server

In the case of the Glassfish server WebSocket and SSE technologies obtain very similar results. For small sizes of messages transmission sequences in LAN achieve shorter times than in WAN. For sizes 7500 ÷ 10,000 bytes the global network begins to gain an advantage, regardless the browser is used. Such behaviour may be caused by the fact that both the client and the server in LAN use the same WiFi access point (the device must thus perform a double job).

3.3 Bidirectional Communication

Bidirectional communication module of performance measurement tool is used to compare a set of Server-Sent Events + XMLHttpeRequest technologies and WebSocket technology. In the first stage of research the technologies were tested

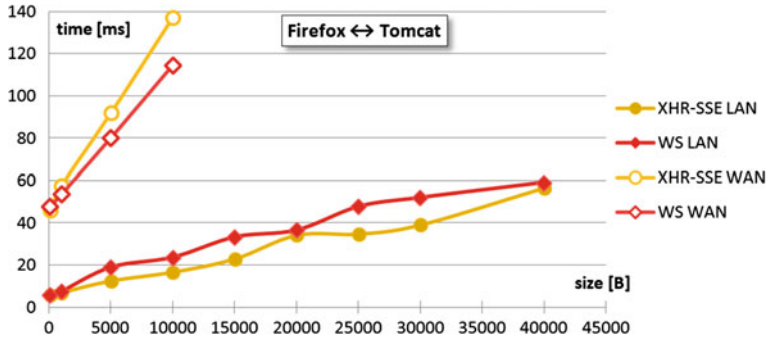

Fig. 8 The time of message pair transmission depending on its size for Firefox browser and the Tomcat server

using the local area network. In the second phase, during the tests which required the use of the global network, there was a problem with the cumulative delay in the transmission of pairs of messages between endpoints. For example, the sequence lasting 16 s in the local network lasted 137 in the global network. For these reasons, the research on the WAN was done only for short messages (the first four of the nine sizes of messages).

Figure 8 presents the comparison of the average message transmission time between Firefox browser and the Tomcat server for the technologies tested. It can be seen that the global network transmissions exhibit similar results but shorter transmission times of message sequences are obtained by WebSocket technology. Such a result was also observed in the case of the other browsers. Since the trend is of linear character, the gain in transmission time for the sequence of messages with an increase in the size of the message can be determined. For WebSocket technology the gain is approximately 6.8 ms per 1000 bytes and for SSE + XHR 8.8 ms.

In the case of a local network it can be seen that for small message sizes (100 or 1000 bytes) HTTP header size in XHR technology has no significant effect on message time transmission, as occurred in the case of transmission from the client to the server. Considering messages of 100 bytes and the average results for the tested browsers, a set of SSE + XHR technologies results in about 15 % longer transmission time sequence of messages in relation to the WebSocket technology. In contrast, in the case of communication from the client to the server, XHR times were 25 times longer as compared to WebSocket for messages of 100 bytes.

Figure 9 shows the comparison of the efficiency of transmission between Web browsers that use WebSocket technology and the Tomcat server. It can be seen that the shortest times for the majority of the message sizes surveyed are reached by Opera. The last on the list is Firefox browser, which, in almost every aspect, is slower than Opera.

In the case of SSE + XHR technologies the results are opposite to WebSocket technology. Firefox, for SSE + XHR technology achieves the shortest time for the majority of message sizes surveyed.

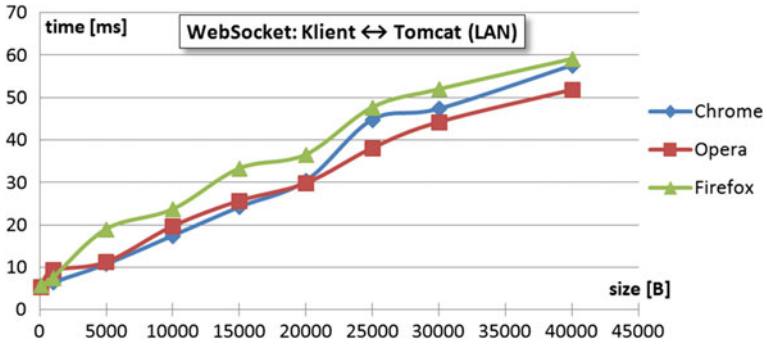

Fig. 9 Message pair transmission time via WebSocket technology, depending on its size for each browser tested, LAN, Tomcat server

3.4 Discussion of Research Results

The problem of technology selection taking into account message sending times only is not trivial. There are many factors influencing the performance of the technology and the ability to use it in a web application. The developers of an application may influence some of them, such as the selection of a server and network configuration, but not all the factors are dependent on them. For this reason, the performance of studies of a similar nature, as those performed in this section, has a positive influence on the decisions people who design applications make and the perception of application users.

In the tests that were carried out the differences in the performance of the technologies could be observed repeatedly depending on the browser used in the studies. It is likely that browsers have different mechanisms implemented to support the operation of web applications. For example, at the expense of optimization of request types that were not analyzed in the study, such as an HTTP request using the cache, or encrypted connections, browsers may worsen the times of simple requests, such as those carried out during the tests. There are many planes of web browsers' activity that may have direct impact on the transmission times achieved and which are not affected by application developers. Such elements may include, for example, JavaScript code interpreter or network sockets manager.

In the course of the research, certain issues which should be considered before performing similar studies were noticed. The first one is hardware. The computer, which worked as a server, turned out to be too weak to generate random messages of such large size, so the results of the tests for communication from the server to the client were disrupted. The second issue concerns the two-way communication module. While the module performs correctly when communicating on LAN, the times obtained in the case of the global network are disrupted by the accumulating communication delay, as a result of which both the server and the client browser

have downtimes at the time of data transmission over the network. All the tested technologies were used in one-way communication modules, so a two-way module may be considered redundant and abandoned in the future.

4 Summary

This chapter was concerned with the analysis of the performance of Web-based systems using XMLHttpRequest, Server-Sent Events and WebSocket. The problem of the performance and capabilities of web technologies is not an exhausted subject, as the above survey, as well as those described in the literature, are of great importance to the development and the future of web applications.

The important conclusion from the study is that the issue of web technology selection is complicated and must not be reduced merely to the choice of the fastest technology. There are many factors that web application developers should consider before choosing a communication technology. On the basis of performance tests, it may be concluded that the best technology in terms of speed in bidirectional communication is WebSocket. If a Web application uses only the communication from the client to the server, then, in terms of speed, XMLHttpRequest technology is slower than WebSocket, but XHR technology advantages associated with the HTTP protocol can be essential for the application. In the case of communication from the server to the client SSE and WebSocket technologies show similar performance. In this case, other factors such as network configuration and browser support should be considered.

To thoroughly examine the technologies listed, such improvements as research time, test points, the other server applications or hardware should be considered. All the tested technologies are constantly improved by the developers of web browsers and the organizations responsible for creating characteristics, so the results obtained from similar studies may become obsolete over time.

Among the directions for further research the following may be listed, for example: enhanced performance testing, testing with other criteria, following the development of current technologies and finding new solutions.

References

1. Dyck, J., Gutwin, C., Graham, N., Pinelle, D.: Beyond the LAN: Techniques from network games for improving groupware performance. In: Proceedings of the ACM Conference on Organizational Computing and Groupware Technologies, pp. 291–300 (2007)
2. Gutwin, C.A., Fedak, C., Watson, M., Dyck, J., Bell, T.: Improving network efficiency in real-time groupware with general message compression. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work CSCW'06, pp. 119–128 (2006)
3. Cook, D.: Data Push Apps with HTML5 SSE. O'Reilly Media, Sebastopol (2014)

4. Grigorik, I.: High-Performance Browser Networking. O'Reilly Media, Sebastopol (2013)
5. Gutwin C.A., Lippold M., Graham N.: Real-time groupware in the browser: Testing the performance of web-based networking. In: Proceedings of the 2011 ACM Conference on Computer Supported Cooperative Work CSCW'11, pp. 167–176 (2011)
6. XMLHttpRequest Level 1: W3C working draft 30 Jan 2014. http://www.w3.org/TR/XMLHttpRequest/
7. Server-Sent Events: W3C working draft 20 Oct 2011. http://www.w3.org/TR/2011/WD-eventsource-20111020/
8. The WebSocket Protocol: Request for comments: 6455. Dec 2011. https://tools.ietf.org/html/rfc6455
9. The Web Sockets API: W3C working draft 22 Dec 2009. http://www.w3.org/TR/2009/WD-websockets-20091222/

Part II
Computer Networks and Distributed Computing

Neighbor Discovery++: A Low-Overhead Address Auto-configuration to Enable Robust Internet of Things Architectures

Monika Grajzer and Mariusz Głąbowski

Abstract With the approaching vision of the Internet of Things smart devices tend to be exploited in a variety of business processes and applications. In order to serve their demands, it is necessary to ensure self-configuration, self-adaptation and self-management capabilities. The Internet Protocol version 6 (IPv6) is a good basis for the networking layer of the Internet of Things. However, its internal mechanisms often fail in ad hoc, mobile set-ups which are common among smart devices. To address these challenges in the field of stateless address auto-configuration, we have proposed an IPv6 Neighbor Discovery++ protocol. In this article we present the protocol overhead evaluation in the set-ups allowing to achieve duplication detection reliability close to 100 %. The results reveal that only a few messages per node are sufficient to verify address uniqueness in diversified network environments. Hence, the solution can serve the requirements of robust Internet of Things architectures.

Keywords Neighbor Discovery · Internet of Things · Address auto-configuration · Duplicate address detection · Duplication detection performance · Neighbor Discovery++

1 Introduction

An increased attention is being put recently towards the Internet of Things (IoT) technologies. The growing number of smart devices, such as mobile terminals, beacons, sensors, actuators, wearables, is being exploited in the variety of business processes—in freight transport, logistics, intelligent offices and buildings,

M. Grajzer
Gido Labs sp. z o.o., Poznań, Poland

M. Grajzer (✉) · M. Głąbowski
Poznań University of Technology, Poznań, Poland
e-mail: monika.grajzer@gidolabs.eu

M. Głąbowski
e-mail: mariusz.glabowski@put.poznan.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_7

personalized shopping, advertising and more. It is envisioned [1, 2] that smart devices will have extended capabilities and will become more autonomic in the future. They will take advantage from the self-configuration, self-adaptation and self-management capabilities [2, 3]. Thus, they will need to communicate freely among themselves to allow for cooperative actions as well as have global connectivity in order to work with cloud storage and other external resources. Autonomy, however, brings also new challenges—while the devices become less isolated, the networks they create call for more robustness.

The Internet Protocol version 6 (IPv6) is a good basis for the networking layer of the Internet of Things. Nevertheless, its targeted scope is within the fixed networks environment, whereas IoT, especially in the access network part, calls for mobile, ad hoc structures. Thus, it is better described as a mobile ad hoc network (MANET) environment. As a result, many challenges arise from the incompatibility of some of the IPv6 solutions with the MANET set-ups which are characterized by dynamic changes, lack of a pre-established infrastructure and strong reconfiguration needs.

Especially self-configuration capabilities of IoT networks are of significant importance, since the connected devices are supposed to interact and create rapid infra-structures in a "plug-and-play" manner [2], without the need for a willful actions of their users and supervising operators. In particular, the aspects of address auto-configuration (AAC) become crucial in this context. Owing to the fact that 50 billions of interconnected devices are prognosticated by Cisco for the IoT in 2020 [4], the assignment of a unique address to the network nodes is not straightforward. In the demanding environment of future innovative networks each new IPv6 address has to be verified [5, 6], since the duplications, even between more distant nodes, can be very harmful for the whole network configuration. Thus, the IETF RFCs require to verify each newly assigned address [7], which is strongly related to the recent common duplications between MAC interface cards addresses being traditionally used as a basis for creating unique node identifiers. The reliability of the duplicate address detection procedure is thus of significant importance and should be duly considered.

The stateless address auto-configuration (SAA) procedures are particularly suitable to address the needs of robust mobile ad hoc set-ups for the IoT, because they provide a mechanism for configuring a valid and unique (not duplicated) IPv6 addresses without the need for external configuration servers, which are often either not present or inaccessible. However, the mechanisms proposed for SAA in IPv6 are not able to address the needs of MANET-like networks, since the duplicate address detection (DAD), being the core part of the whole procedure, cannot be performed between more distant nodes. This is acceptable in fixed networks, but can cause severe issues in networks where the configuration constantly changes, nodes move, connect and disconnect. Therefore, new control and configuration solutions are needed to extend IPv6 [5].

Current stateless address auto-configuration solutions for MANETs [8–12] are not sufficient to address the above issues, since they lack adequate robustness and efficiency. The limitations of these approaches are particularly visible for large-scale

networks, where the protocol overhead issues become a substantial factor influencing whole network operation.

To address the above challenges we have proposed a method of efficient DAD as an extension to one of the core IPv6 protocols—the Neighbor Discovery (ND) protocol [7, 13]. The proposed Neighbor Discovery++ (ND++) [14–16] provides enhanced address duplication capabilities enabling to verify address uniqueness between more distant nodes. Moreover, ND++ is capable of reacting to network changes and keeping protocol overhead low, even in demanding networking conditions. ND++ most suitable set-up, which was identified in the course of our previous research [17, 18], allows to achieve duplication detection probability at the levels of 90 % and above, often close to 100 %. In this article we present the evaluation of this set-up from the protocol overhead perspective. The findings reveal that the overhead can be kept at very low levels for the variety of network sizes, especially in large-scale networks. The proposed solution can be exploited independently of other protocols within protocol stack, such as routing, MAC layer protocols, application layer mechanisms (MQTT, CoAP, etc.). Those features make ND++ an interesting solution focused on the interconnection of significant number of smart devices into the Internet of Things.

This article is organized as follows: Sect. 2 presents the related work and elaborates on the differences with our proposed solution, Sect. 3 describes key mechanisms and features of ND++ protocol, whereas Sect. 4 evaluates its performance based on the simulation results. Finally Sect. 5 concludes the paper.

2 Related Work

SAA approach for MANET networks provides means to verify address uniqueness in a networks of limited initial configuration available. There are several approaches proposed to address related issues [6]. The so called “strong DAD” aims to verify address uniqueness by sending address queries. If a reply is not received within a specified amount of time, the address is considered unique. On the other hand “weak DAD” and “passive DAD” are oriented towards inferring through the network behavior observations that the duplications exist. SAA based on the “strong DAD” approach maximizes the reliability of duplication detection. Moreover, it is similar to the method for SAA incorporated as a part of intrinsic IPv6 mechanisms within the Neighbor Discovery (ND) protocol [7, 13]. Therefore we also follow this approach. Considering our proposed solution—the ND++ protocol and, in particular, the aspects of its overhead, we depict below the most relevant related works referring to the ND protocol.

Concerning the IPv6 ND protocol [13] and the IPv6 SAA procedure [7], they were both designed for fixed network environments and therefore are limited to the closest, link-local neighborhood of each node. Hence, they verify address uniqueness by sending Neighbor Solicitation address queries only to the directly connected nodes (forwarding is not allowed), thus by design permitting more

distant address duplications. Such a situation is not acceptable in the investigated target environment of IoT. Therefore, ND++ has wider range covering possibly a whole ad hoc network domain.

Increased coverage is also the key feature of the solutions proposed so far particularly for MANET network environments [8–12]. It is achieved by allowing to forward ND protocol messages, in the extended range of *n* hops. However, as shown by our previous research [16], unrestricted flooding of multihop protocol messages, such as e.g. in [11], will likely lead to broadcast storm. It can paralyze network functionality, especially in large networks. However, flooding restriction is not straightforward in case of auto-configuration protocol—there is often no valid address assigned to the network interface of a device before the address configuration functionality is finished, and thus communication with other nodes cannot be started. Hence, all methods, which rely on the initial wide topology information (such as e.g. MDR algorithm [19]), cannot be used. Due to this fact, previous solutions either have not restricted flooding at all [11], or have tried to minimize the scope in which unrestricted flooding is performed by creating a hierarchical structure within a network on the additional cost of leader selection process [8–10]. In general, the reference solutions designed for MANET networks are characterized by relatively high protocol overhead and are not always capable of being easily deployed with other commonly used IPv6 protocols. Our solution, ND++, can significantly reduce protocol overhead without the need for external configuration nodes. It is achieved through the proposed algorithmic procedure, which enabled to use Multipoint Relay (MPR) flooding optimization even before the AAC processes are fully finalized. Moreover, ND++ is fully compliant with IPv6 standards and backward compatibility with ND [13] is ensured, as required by [5].

Among investigated previous works there were also attempts to shift the paradigm and include ND functionality in some of the routing protocols (e.g. OLSR, which also uses MPR mechanism). However, it is argued [5, 6] that routing and AAC should be independent processes and we follow this approach.

3 Neighbor Discovery++ for Enhanced Duplicate Address Detection and Address Auto-configuration

In the basic ND solution [7, 13] DAD is performed by sending a query for the address of interest within a specified scope by means of a multicast Neighbor Solicitation (NS) messages. If another node in the network already owns this address, it sends a reply and the address cannot be used. In case it is not received within a specified time, the address is considered valid. ND++ protocol proposes several extensions to basic ND which are aimed to allow for efficient duplicate address detection in mobile, ad hoc network environments.

First of them is the extended range, which allows to verify address uniqueness in the scope of *n* hops, instead of just 1-hop neighborhood of each node. It is achieved

by allowing to forward NS address queries through several hops specified by the *HopLimit* protocol parameter, which specifies the protocols range.

Second ND++ extension ensures protocol overhead control. In ND++ we have proposed to exploit the Multipoint Relay (MPR) mechanism from the OLSR protocol [20] for efficient flooding of multihop protocol messages. Thanks to MPR mechanism capabilities, it can be incorporated at the early network operation phases, such as in case of SAA procedures, since it does not require network-wide topology information. However, in order to be able to integrate it into ND++ functionality at the stage where the investigated IPv6 address is not completely verified for uniqueness yet (and thus cannot be used), we have proposed a modification in the core ND algorithmic procedure. Hence, in ND++ DAD is performed not in one, but in two stages: first in the range of 1-hop, similarly as in the standard ND, second in the extended range of n hops (n-DAD), which allows to cover a whole MANET domain while using slightly modified basic protocol messages, which can be forwarded. The whole process is denoted as DAD++.

In parallel to the exchange of NS messages and to the DAD++ procedure, the MPR processes are operational. Each node in the network collects information about its 2-hop neighborhood by exchanging link-local messages (having 1-hop scope) with its direct neighbors [14, 15]. The frequency of sending MPR messages is a configurable parameter—in the current ND++ set-up it is 1 message per second. Based on the collected data each node chooses MPRs—nodes willing to forward ND ++ information on its behalf, following the heuristic defined in [14, 15, 20]. This heuristic allows to specify a subset of direct neighbors which will allow to reach all 2-hop neighbors of a node. Having an address verified as unique in the range of 1-hop (after completion of first DAD++ stage) is sufficient to perform both this kind of information collection and MPR selection process. Hence, the MPRs are chosen for each node before the second DAD++ stage and they can be used for forwarding packets from their MPR selectors during n-DAD, which exploits multihop NS (mNS) messages.

Our recent results [17, 18] revealed that the forwarding through MPRs should be restricted to one copy of each message. This is achieved by monitoring the parameters of previously forwarded messages. This way the protocol overhead can be kept constant with the increased *HopLimit* value—after reaching the value large enough to cover all nodes in the network, the overhead is not increased any further [18]. This feature is very important from the performance perspective. It also allows for straightforward set-up of the *HopLimit* parameter, which can be chosen even as equal to the expected node count.

The third feature applied to ND++ enables to maximize protocol reliability in realistic environments. Flooding control allows for the minimization of a protocol overhead, but on the other hand makes the protocol less resistant to packet loss. This results in the decreased probability of successful duplication detection. In order to deal with this issue, we have proposed several remedy approaches [17, 18], from which the one allowing for cyclic n-DAD queries was identified as the best considering both—the duplication detection probability and the imposed overhead. In this approach the second DAD++ stage is repeated more than once (2 or 3 times).

What is also compelling is that ND++, similarly to ND and SAA described in [7], is aimed mainly at verification of address uniqueness (DAD). Therefore, it is envisioned that in order to complete SAA and provide each node with a routable IPv6 address, ND++ will be complemented by some additional prefix assignment mechanism. There can be diversified mechanisms used for this purpose with ND++ as a core technology. These are out of the scope of this article.

4 Performance Evaluation

Enhancing ND++ reliability (specified as the probability of successful duplication detection) through the incorporation of cyclic *n*-DAD queries is provided on the additional cost of an increased protocol overhead. As discussed in detail in [18], this methodology allows to reduce the overhead imposed by additional *n*-DAD queries in case of a duplication in a network—once it is detected, the scheduled *n*-DAD queries, which have not been started yet, are cancelled. However, in case all addresses are unique, all of the planned queries will be executed. Owing to the fact that most likely the lack of address duplications could be more probable in many network set-ups, it is important to assess ND++ performance in terms of overhead in both scenarios (with and without duplication in a network). Hence, in this article we present a detailed analysis of the ND++ overhead in the protocol set-up allowing for achieving reliability between 90 and 100 %. This study enables to specify upper overhead boundaries in diversified network situations.

Moreover, since the increased number of *n*-DAD cycles leads to better probability of successful duplication detection, it may be beneficial to repeat it more times, than the required minimum, to accommodate for more demanding networking environments. This is driven by the fact that our initial study presented in [18] revealed that the minimum required *n*-DAD repetition count is dependent on the topology. Hence, we also present the comparative study showing the ND++ overhead in two cases: (1) *n*-DAD is repeated a minimum number of times, which allows to achieve reliability of 90 % and above; (2) *n*-DAD is performed an additional time more.

4.1 Simulation Environment

The ND++ protocol evaluation was performed in the well-established NS-3 simulation environment [21], which has very good IPv6 stack implementation (for further details on the simulation environment selection please refer to [22]). The simulated MANET network was specified as an 802.11-based, IPv6-only set-up (OFDM mode, 54 Mbps rate to allow for a maximum throughput) with a variable number of nodes (16, 36 or 100 nodes). The channel is modeled by means of a log-distance path loss model. Fading channels were not taken into

consideration, since they incorporate also gain which leads to topology changes. Our objective is, however, to keep topologies as comparable as possible. Moreover, our investigations have shown that the impact of additional fading channel model on the results was negligible.

Three different topologies have been investigated: 2 predefined grid topologies—the variant without diagonal connections (denoted as Grid) and with those connections included (Cross-Grid) as well as a random node distribution on the disc area (Uniform Disc), which is a good representative of random topologies. The grid-based topologies were selected due to their deterministic characteristics allowing for scaling them and thus for comparing results obtained for different number of nodes with a similar layout.

During the experiments the 95 % confidence intervals of the t-Student distribution were taken into consideration—the simulation run was terminated when the half of the confidence interval length was within 10 % of the estimated average value. Initially 10 data points were collected for each averaged value, but this number was increased if necessary. However, for some of the topologies, in the case with duplication present, the overhead values may vary significantly when the duplication is detected fast and the remaining *n*-DAD trials are neglected. In such a situation the accuracy control, as defined above, could not be operational, since the diversity of the results in principle was too high. In these cases 70 trials were performed and if at that point the convergence could not have been reached, the procedure was stopped. We believe 70 trials are more than enough to provide a good estimation of the final values.

4.2 Results

The experiments were aimed to measure a number of mNS messages (i.e. the messages from the *n*-DAD phase) generated in the entire network during a full DAD++ query for a single new IPv6 address as a function of the *HopLimit* parameter. Two variants were considered—the duplication was either present or not. In case of scenarios with an existing (single) duplication, the node initiating the process, which assigned itself a new address, and the target node possessing the address of interest were chosen so that the distance between them is possibly large. The same initiating node was chosen in both investigated scenarios.

The ND++ parameters are set so that: RETRANS_TIMER value (specifying a timeout to wait for a reply after sending an address query in case of both DAD and *n*-DAD) is 1s (for 16 and 36 node network; 1.5s for 100 node network), MPR_DELAY (time between consecutive MPR information announcement events) is 1s and NDAD_DELAY (time to wait between DAD and *n*-DAD) is 3s.

During results comparison the stable values, obtained after exceeding some *HopLimit* threshold, should be considered—these reflect the protocol overhead in a stable-state ND++ set-up.

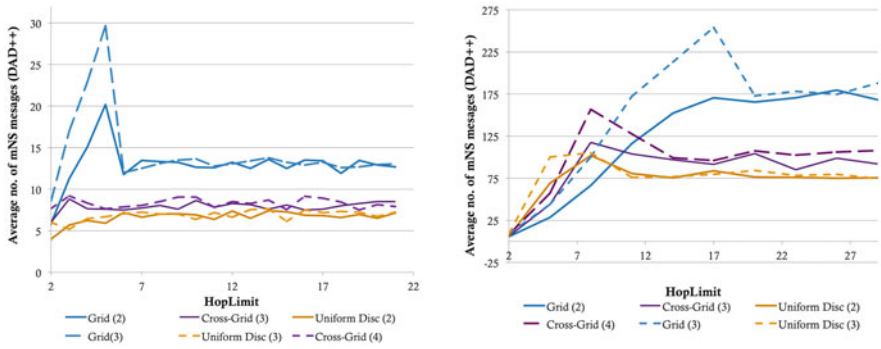

Fig. 1 Duplicated address scenario: number of *mNS* messages sent during a single DAD++ query for a duplicated address versus *HopLimit* value in a network of 16 (*left*) and 100 nodes (*right*)

Figure 1 presents the ND++ performance results in a DAD++ query being performed for a duplicated address. Two different ND++ set-ups were considered.

In the first one, with the results marked with solid lines, the n-DAD is repeated 2 times for Grid and Uniform Disc topology, 3 times—for Cross-Grid topology. Such settings are the minimum *n*-DAD repetition count which allows for achieving reliability exceeding 90 % in each case [18]. It can be observed that maximum protocol overhead for a 16-node network was approximately 13 messages for the Grid topology, 8 and 9 for the Cross-Grid and Uniform Disc node layouts, respectively. The corresponding values for a 100-node network are approximately 175, 95 and 75 messages.

The second set-up depicted in Fig. 1 reflects the situation in which minimal *n*-DAD repetition count for each case was increased by one. The purpose of this investigation was to verify the impact of potentially overestimated repetition counts on the protocol overhead. Interestingly, looking at the stable-state values, there is almost no increase in the overhead observed—this is due to the fact that after successful duplication detection consecutive *n*-DAD trails are suppressed. However, for the *HopLimit* values close to the threshold allowing for full network coverage the situation is different. There are overhead peaks observed in such a case. When the *HopLimit* is relatively small, close to the threshold, it may happen that shorter paths between DAD++ initiating and target node allow to detect a duplication, however ND++ messages travelling on longer paths cannot reach the target yet. Hence, under such conditions the ND++ is more vulnerable to packet loss and, as a result, more *n*-DAD trials may be needed to verify a new address as duplicated. Each trial results in overhead increase, which is visible in the results. However, it is important to notice that the suggested ND++ set-up should use *HopLimit* values significantly larger than the indicated threshold. Thus, the protocol's operation point should be set within the stable overhead values.

Figure 2 presents similar study performed for a scenario where a newly assigned address is unique in the network, i.e. no duplication is present. In this situation the

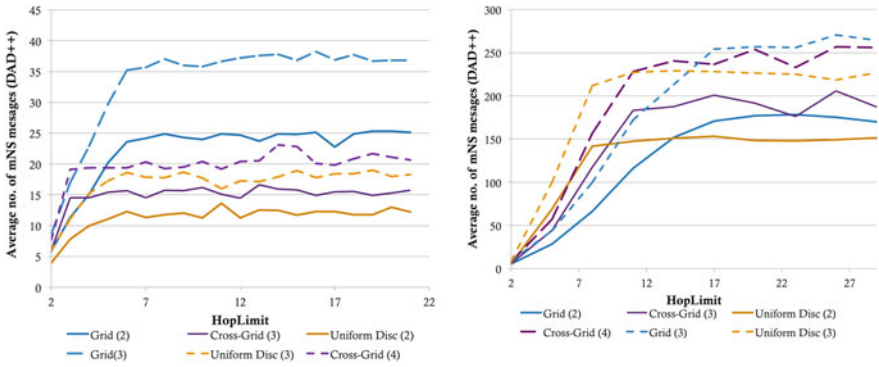

Fig. 2 Unique address scenario: number of *mNS* messages sent during single DAD++ query for a unique address versus *HopLimit* value in a network of 16 (*left*) and 100 nodes (*right*)

difference between both investigated cases appeared—higher *n*-DAD repetition count always led to the overhead increase, since the blocking mechanism, described above, is not possible without the detection of duplication. For the same reason, in both cases (marked with solid and dashed lines) the results are higher than for a duplicated address scenario. Moreover, this aspect implies also that the peak values, observed in Fig. 1 are not present in this case. The results present that the maximum ND++ overhead for the first case set-up, specified with the minimum *n*-DAD repetition count, was at most 25 messages in a 16-node network for a Grid topology and 190 for 100 nodes in a Cross-Grid topology. The evaluation of a second case set-up (with *n*-DAD repetition count increased by one) gave at most approx. 37 messages in a 16-node network for a Grid topology and 260 for 100 nodes in a Cross-Grid topology.

As such, in the worst of the investigated cases the ND++ overhead was approximately 2.6 messages per node, which is a very good result. Moreover, the worst-case values correspond to the ND++ set-up with overestimated *n*-DAD repetition count. For the minimum required set-up the maximum overhead generated in the more difficult scenario, without address duplications in a network, was less than 2 messages per node.

5 Conclusions

The evaluation of ND++ performance in a set-ups, which allow for reaching protocol reliability higher than 90 %, has shown that the protocol overhead is in general in the order of 0.5–3 messages per node, depending on the topology and ND++ set-up. Different situations have been considered for the needs of protocol overhead estimation, including both the presence and lack of duplications in the network. In both scenarios, even in large-scale networks, the overhead was kept

very low. Moreover, the comparison between minimal and optimal *n*-DAD repetition count in ND++ set-up has shown that the results are similar in case of existing duplications, however, the overhead remains still low if the addresses are unique. Therefore, it seems that in most network set-ups overestimating *n*-DAD repetition count should not be harmful to the network in terms of overhead, but could be beneficial in terms of increased reliability. Additionally, the *n*-DAD repetition count is a configurable protocol parameter, so it could be adjusted to the needs of particular network environments. Hence, it seems that ND++ has a required flexibility, reliability and achievable performance to serve as a good basis for the future auto-configuration services. With the increased SAA capabilities future networks, which can address the requirements of demanding Internet of Things architectures, technologies and applications, become realizable and closer to practical implementations.

Acknowledgments Monika Grajzer is a scholarship holder within the project "Scholarship support for Ph.D. students specializing in majors strategic for Wielkopolska's development", Sub-measure 8.2.2 Human Capital Operational Programme, co-financed by European Union under the European Social Fund. The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task "Struktura, analiza i projektowanie nowoczesnych systemów komutacyjnych i sieci telekomunikacyjnych" in 2015.

References

1. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of Things: vision, applications and research challenges. Ad Hoc Netw. **10**(7), 1497–1516 (2012)
2. OECD: Machine-to-Machine Communications. Connecting Billions of Devices. OECD Digital Economy Papers, No. 192, OECD Publishing (2012)
3. Lake, D., Rayes, A., Morrow, M.: The internet of things. Internet Protoc. J. **15**. (2012). Chief Technology Office, Cisco Systems, Inc
4. CISCO website: Internet of Things (IoT). http://www.cisco.com/web/solutions/trends/iot/portfolio.html. Accessed on 4 June 2015
5. Baccelli, E.: Address autoconfiguration for MANET: terminology and problem statement draft-ietf-autoconf-statement-04. IETF Internet Draft (Work in progress, Expired), Feb 2008. https://tools.ietf.org/html/draft-ietf-autoconf-statement-04
6. Garcìa Villalba, L.J., Garcìa Matesanz, J., Sandoval Orozco, A.L., Márquez Dìaz, J.D.: Auto-configuration protocols in mobile ad hoc networks. Sensors **11**(4), 3652–3666 (2011)
7. Thomson, S., Narten, T., Jinmei, T.: RFC4862: IPv6 stateless address autoconfiguration. IETF Draft Standard, Sept 2007. http://www.rfc-editor.org/rfc/rfc4862.txt
8. Weniger, K., Zitterbart, M.: IPv6 autoconfiguration in large scale mobile ad-hoc networks. Proc. Eur. Wireless. **1**, 142–148 (2002)
9. Weniger, K., Zitterbart, M.: IPv6 stateless address autoconfiguration for hierarchical mobile ad hoc networks. IETF Internet Draft (Work in progress, Expired), Feb 2002. http://tools.ietf.org/id/draft-weniger-manet-addressautoconf-ipv6-00.txt
10. Wang, X., Qian, H.: Dynamic and hierarchical IPv6 address configuration for a mobile ad hoc network. Int. J. Commun Syst **28**(1), 127–146 (2015)
11. Park, J., Kim, Y., Park, S.: Stateless address autoconfiguration in mobile ad hoc networks using site-local address. IETF Internet Draft (Work in progress, Expired), July 2001. http://tools.ietf.org/id/draft-park-zeroconf-manet-ipv6

12. Boudjit, S., Laouiti, A., Muhlethaler, P., Adjih, C.: Duplicate address detection and autoconfiguration in OLSR. J. Univers. Comput. Sci. **13**(1), 4–31 (2007)
13. Narten, T., Nordmark, E., Simpson, W., Soliman, H.: RFC4861: neighbor discovery for IP version 6 (IPv6). IETF Draft Standard, Sept 2007. http://www.rfc-editor.org/rfc/rfc4861.txt
14. Grajzer, M.: ND++—an extended IPv6 Neighbor Discovery protocol for enhanced duplicate address detection to support stateless address auto-configuration in IPv6 mobile ad hoc networks. IETF Internet Draft (Work in progress), Mar 2011. http://tools.ietf.org/html/draft-grajzer-autoconf-ndpp-00.txt
15. Grajzer, M., Żernicki, T., Głąbowski, M.: ND++—an extended IPv6 Neighbor Discovery protocol for enhanced stateless address autoconfiguration in MANETs. Int. J. Commun Syst **27**(10), 2269–2288 (2014)
16. Grajzer, M., Głąbowski, M.: Performance evaluation of Neighbor Discovery++ protocol for the provisioning of self-configuration services in IPv6 mobile ad hoc networks. In: IEEE 16th International Telecommunications Network Strategy and Planning Symposium (Networks), 2014, pp. 1–6
17. Grajzer, M., Głąbowski, M.: On the probability of duplicate address detection with Neighbor Discovery++ protocol in IPv6 mobile ad hoc networks. In: Proceedings of the 2015 IEICE General Conference, ISSN 1349-1377, Japan, 10–13 Mar 2015
18. Grajzer, M., Głąbowski, M.: On the reliability of duplicate address detection in mobile ad hoc networks with Neighbor Discovery++ address auto-configuration protocol. In: Proceedings of the IEICE Information and Communication Technology Forum 2015, Accepted paper, Manchester, UK, 3–5 June 2015
19. Ogier, R., Spagnolo, P.: RFC 5614: mobile ad hoc network (MANET) extension of OSPF using connected dominating set (CDS) flooding. IETF Draft Standard, Aug 2009. http://wiki.tools.ietf.org/html/rfc5614
20. Clausen, T., Jacquet, P.: RFC3626: Optimized Link State Routing protocol (OLSR). IETF Draft Experimental, Oct 2003. http://www.rfc-editor.org/rfc/rfc3626.txt
21. NS-3 simulator website: http://www.nsnam.org
22. Grajzer, M., Głąbowski, M.: On IPv6 experimentation in wireless mobile ad hoc networks. J. Telecommun. Inf. Technol. (JTIT) **3**(2014), 71–81 (2014)

Modeling and Optimization of the Production Cluster

Zainelkhriet Murzabekov, Marek Milosz and Kamshat Tussupova

Abstract The paper presents an economic model of a production cluster (EMPC) consisting of three subsystems: supply of materials and means of labor, as well as production of final product. Authors defined the task of finding EMPC's optimal steady state, given a condition of final product production volume maximization. Authors provide an algorithm for searching production cluster's steady state. The investigation results of an effect of various parameters on EMPC's optimal steady state are presented. The problem of optimal control for production clusters is formulated.

Keywords Production cluster · Three-sector model · System's steady state · Optimal control

1 Introduction

Currently, clustering is one of the priorities in the global economy [1]. Particular interest to cluster systems is associated with trends of grouping and consolidation of capitals, which can lead to activation of production integration processes. During transition (or development) process to the economy based on clusters, the government, realizing its common national goals, stimulates development of clusters in various industries.

Z. Murzabekov (✉) · K. Tussupova
Department of Information Systems, Al-Farabi Kazakh National University, Almaty, Kazakhstan
e-mail: murzabekov-zein@mail.ru

K. Tussupova
e-mail: kamshat-0707@mail.ru

M. Milosz
Institute of Computer Science, Lublin University of Technology, Lublin, Poland
e-mail: marekm@cs.pollub.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_8

The core of effective functioning of a cluster as an economic system is the process of group strategic planning. To support strategic legal and economic decision-making, it is necessary to create and explore economic models of both closed and open production clusters [2–4]. The theory of parametric control of an economy [5] can be used to the government stimulate of production clusters. This paper shows the theoretical and practical fundamentals of those processes.

2 Three-Sector Model of a Cluster

Production cluster can be represented as a set of three interrelated subsystems that use materials (objects of labor), labor resources (labor force) and production assets (means of labor) for production. Subsystem supplying cluster's material delivers material resources for all cluster's subsystems. Subsystem supplying means of labor creates productive assets of all subsystems in the cluster. Production subsystem produces final selling products in the cluster. Thus, the model of production cluster is a three-sector model [6–9].

Figure 1 shows a block diagram of a production cluster with materials and tools flows. When constructing a mathematical model it must take into account, that the consider system is closed [10], in the sense that there is no external constraints on flows: materials, labor, investments and production. This means that demand for production of cluster's final goods is unlimited.

Markings: $\theta_0, \theta_1, \theta_2$—subsystems' shares in distribution of labor resources, s_0, s_1, s_2—subsystems' shares of in distribution of investment resources, $\beta_0, \beta_1, \beta_2$—shares of material costs in subsystems.

It is important that subjects of production cluster cooperate to obtain as much amount of final product as possible. All particular interests of individual subsystems are subordinated to that goal. Thus a production cluster allocates all available resources completely.

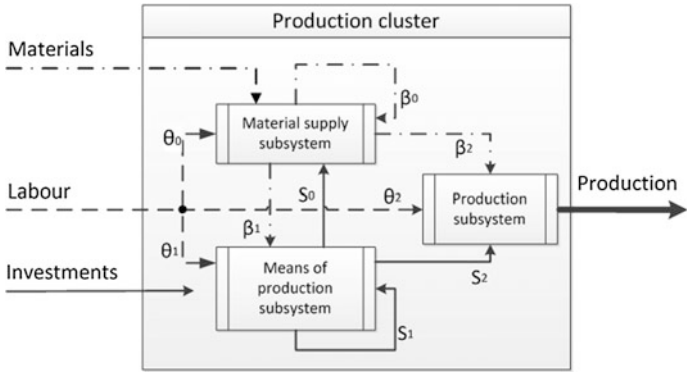

Fig. 1 Structural three-sector model of a production cluster

To build an Economic Model of a Production Cluster (EMPC), i is used to represent subsystems (sectors) of the model, where $i = 0$—material, $i = 1$—creation of funds, and $i = 2$—production.

Economic conditions of EMPC functioning are given by specific coefficients:

- A_i—coefficient of neutral technical progress, it is a factor that commensurate resources with production;
- α_i—elasticity coefficient of funds shows how output increases, if a fund is increased by 1 % (%);
- β_i—direct material costs per output unit of i-th sector (%);
- $\lambda_i = \mu_i + \nu$—capital wear factor of i-th sector (% per year);
- μ_i—proportion of withdrawn fixed assets a year (% per year);
- ν—annual growth rate of employees (% per year).

Technological setup is considered constant and is given by linear homogeneous neoclassical Production Functions (*PF*) [11], which express dependence of production result from resources spent. Using the PF in the model, input resources and output results we can transform to the final products' annual production. As resources, investment capital K and number of employees L are considered, and as a result—production output X. Thus, PF can be in general represented as [11]:

$$X_i = F_i(K_i, L_i), \quad i = 0, 1, 2. \tag{1}$$

It is also assumed that each subsystem has fixed Production Assets (PA), while labor and investments can be moved freely between sectors. Change of PA of i-th sector in the period Δt consists of wearing $(-\mu_i K_i)$ and growth because of investment (I_i), i.e.:

$$K_i(t + \Delta t) - K_i(t) = [-\mu_i K_i(t) + I_i(t)]\Delta t, \quad i = 0, 1, 2, \tag{2}$$

when $\Delta t \to 0$ we receive differential equations for sectors' PA [11]:

$$\frac{dK_i}{dt} = -\mu_i K_i + I_i, \quad K_i(0) = K_i^0, i = 0, 1, 2. \tag{3}$$

The following notation is introduced:

- $\theta_i = \frac{L_i}{L}$—share of sectors in workforce distribution (%);
- $s_i = \frac{I_i}{X_1}$—share of sectors in investment resources distribution (%);
- $f_i(k_i) = \frac{X_i}{L_i}$—labour productivity of i–th sector ($ per person per year);
- $k_i = \frac{K_i}{L_i}$—sectors' capital-labor ratio ($ per person);
- $x_i = \theta_i f_i(k_i)$—sectors' specific output ($ per person per year).

Then Eq. (3) is written in the following form for sectors' capital-labor ratio:

$$\frac{dk_i}{dt} = -\lambda_i k_i + \frac{s_i}{\theta_i} x_1, \quad \lambda_i > 0, k_i(0) = k_i^0, i = 0, 1, 2. \tag{4}$$

Condition of complete allocation of resources in the cluster leads to the following restrictions (work, investment and material balances):

$$\theta_0 + \theta_1 + \theta_2 = 1, \quad 0 < \theta_i < 1, \tag{5}$$

$$s_0 + s_1 + s_2 = 1, \quad 0 < s_i < 1, \tag{6}$$

$$(1 - \beta_0)x_0 = \beta_1 x_1 + \beta_2 x_2, \quad \beta_i > 0. \tag{7}$$

Equations (4)–(7) constitute the overall EMPC. Considering the case where PF is Cobb-Douglas functions [12]:

$$X_i = F_i(K_i, L_i) = A_i K_i^{\alpha_i} L_i^{1-\alpha_i}, \quad i = 0, 1, 2. \tag{8}$$

$$f_i(k_i) = \frac{X_i}{L_i} = A_i k_i^{\alpha_i}, \quad A_i > 0, 0 < \alpha_i < 1, i = 0, 1, 2. \tag{9}$$

Using (9), after appropriate transformation differential equations (4) will be as follows:

$$\frac{dk_i}{dt} = -\lambda_i k_i + \frac{s_i}{\theta_i} \theta_1 A_1 k_1^{\alpha_1}, \quad k_i(0) = k_i^0, i = 0, 1, 2. \tag{10}$$

In the general case constructed model can be represented as in Fig. 2

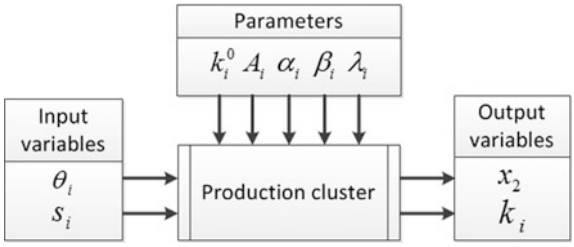

Fig. 2 Input and output parameters of the EMPC

3 Optimal Equilibrium State of a Cluster

State of the production cluster is set by values of resource allocation parameters. The state which does not change the value of sectors' ratio, will be the stationary state. In the set of stationary states of the production cluster, there can be find the optimal one, i.e., in which production of final product is maximized. This leads to following formulation of the problem of finding an optimal state of production cluster.

3.1 Statement of the Problem of Finding the Optimal Steady State

The task of finding the Optimal Steady State of a Production Cluster (OSSPC) is reduced to the problem of finding the point at which the state of EMPC is stationary and the value of specific production of consumer subsystem ($x_2 = \theta_2 A_2 k_2^{\alpha_2}$) is maximal in all of the acceptable ranges of control parameters (θ_i, s_i).

The problem of searching OSSPC for EMPC is reduced to following nonlinear programming problem with x_i as an objective function:

$$x_2 = \theta_2 A_2 k_2^{\alpha_2} \longrightarrow [\theta_i, s_i]\max \tag{11}$$

under conditions: (5), (6) and

$$(1-\beta_0)\theta_0 A_0 k_0^{\alpha_0} = \beta_1 \theta_1 A_1 k_1^{\alpha_1} + \beta_2 \theta_2 A_2 k_2^{\alpha_2}, \quad \beta_i > 0, \tag{12}$$

and the system of differential Eqs. (10).

3.2 Solution to the Problem of OSSPC Search

Since stationary solutions of EMPC are sought, capital-labor ratio of subsystems does not vary over time. From this follows that the rate of subsystems' capital-labor ratio change in Eq. (10) is equal to zero:

$$-\lambda_i k_i + \frac{s_i}{\theta_i}\theta_1 A_1 k_1^{\alpha_1} = 0, \quad i = 0, 1, 2. \tag{13}$$

To solve the problem (11), (5), (6), (12) and (13) Lagrange multipliers of a special kind are used. The function is written in the form:

$$
\begin{aligned}
L(\theta_i, s_i, k_i, c_j) = & -\theta_2 A_2 k_2^{\alpha_2} + c_1(\theta_0 + \theta_1 + \theta_2 - 1) + c_2(s_0 + s_1 + s_2 - 1) \\
& + c_3\left((1-\beta_0)\theta_0 A_0 k_0^{\alpha_0} - \beta_1 \theta_1 A_1 k_1^{\alpha_1} - \beta_2 \theta_2 A_2 k_2^{\alpha_2}\right) \\
& + c_4 k_0\left(-\lambda_0 k_0 + \frac{s_0}{\theta_0}\theta_1 A_1 k_1^{\alpha_1}\right) + c_5 k_1\left(-\lambda_1 k_1 + s_1 A_1 k_1^{\alpha_1}\right) \\
& + c_6 k_2\left(-\lambda_2 k_2 + \frac{s_2}{\theta_2}\theta_1 A_1 k_1^{\alpha_1}\right)
\end{aligned} \tag{14}
$$

where, c_j, $j = 1, \cdots, 6$—Lagrange multipliers.

To investigate the extremum of the function $L(\theta_i, s_i, k_i, c_j)$, necessary conditions for an extremum of the first order are written:

$$\frac{\partial L(\theta_i, s_i, k_i, c_j)}{\partial \theta_i} = 0, \quad i = 0, 1, 2, \tag{15}$$

$$\frac{\partial L(\theta_i, s_i, k_i, c_j)}{\partial s_i} = 0, \quad i = 0, 1, 2, \tag{16}$$

$$\frac{\partial L(\theta_i, s_i, k_i, c_j)}{\partial k_i} = 0, \quad i = 0, 1, 2. \tag{17}$$

Thus, from the nonlinear system of fifteen Eqs. (5), (6), (12), (13) and (15)–(17) with fifteen unknown parameters can be calculated using algebraic manipulations.

Algorithm of obtaining OSSPC's final solution begins with finding the value of parameter s_1 from the equation obtained in the form:

$$
\begin{aligned}
& (1-\alpha_0)(1-\beta_0)A_0\gamma_0^{\alpha_0} s_1 + (1-\alpha_1)\beta_1 A_1 \gamma_1^{\alpha_1} s_1^{\frac{1-\alpha_0}{1-\alpha_1}} \\
& - \alpha_1(1-\alpha_0)(1-\beta_0)A_0\gamma_0^{\alpha_0} = 0
\end{aligned} \tag{18}
$$

where γ_0, γ_1 represent following values:

$$\gamma_0 = \frac{s_0\theta_1}{s_1\theta_0}\frac{A_1^{\frac{1}{1-\alpha_1}}}{\lambda_0\lambda_1^{\frac{\alpha_1}{1-\alpha_1}}} = \frac{\alpha_0(1-\alpha_1)}{\alpha_1(1-\alpha_0)}\frac{A_1^{\frac{1}{1-\alpha_1}}}{\lambda_0\lambda_1^{\frac{\alpha_1}{1-\alpha_1}}}, \tag{19}$$

$$\gamma_1 = \left(\frac{A_1}{\lambda_1}\right)^{\frac{1}{1-\alpha_1}}. \tag{20}$$

Using obtained value of the parameter s_1 the rest of unknown variables can be found. All calculations are implemented using the Maple software package.

4 Parametrical Analysis of Balance State

Parametric analysis of change in production cluster's steady state is implemented for exemplary model coefficients. Their values are given in Table 1.

Parametric analysis of EMPC is conducted to investigate the influence of parameters on OSSPC.

As an example, there can be considered two cases that are characterized by changes in the following parameters:

1. Funds elasticity in material supply subsystem (it is displayed by parameter α_0).
2. Direct material costs in production subsystem (it is displayed by parameter β_2).

OSSPC calculation results are given for the case 1 in Table 2 and for the case 2 in Table 3.

Research of EMPC optimal steady state with changing parameters of funds elasticity in the material supply subsystem and material costs of production

Table 1 Initial parameters of EMPC

i	0	1	2
α_i	0.5	0.7	0.5
β_i	0.2	0.3	0.5
λ_i	0.05	0.05	0.05
A_i	5	2	3

Table 2 Calculations of the model when changing the parameter α_0

α_0	x_2	θ_i			s_i		
		0	1	2	0	1	2
0.2	10.35	0.79	0.04	0.17	0.43	0.19	0.38
0.3	17.33	0.71	0.07	0.22	0.45	0.23	0.32
0.4	32.79	0.60	0.12	0.28	0.42	0.29	0.29
0.5	66.54	0.43	0.21	0.36	0.34	0.38	0.28
0.6	126.39	0.23	0.34	0.43	0.22	0.50	0.28
0.7	191.52	0.08	0.44	0.48	0.11	0.61	0.28
0.8	230.69	0.02	0.49	0.49	0.04	0.67	0.29

Table 3 Calculations of the model when changing the parameter β_2

β_2	x_2	θ_i			s_i		
		0	1	2	0	1	2
0.2	79.56	0.36	0.21	0.43	0.28	0.38	0.34
0.3	74.69	0.38	0.21	0.41	0.30	0.38	0.32
0.4	70.38	0.41	0.21	0.38	0.32	0.38	0.30
0.5	66.54	0.43	0.21	0.36	0.34	0.38	0.28
0.6	63.10	0.45	0.21	0.34	0.35	0.38	0.27
0.7	60.00	0.46	0.21	0.33	0.36	0.38	0.26
0.8	57.18	0.48	0.21	0.31	0.38	0.38	0.24

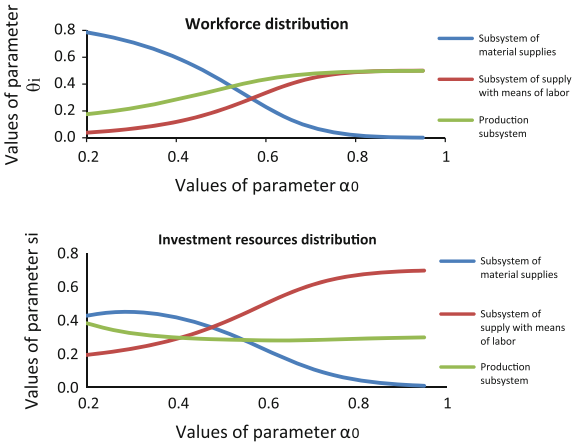

Fig. 3 Effect of changes in the elasticity parameter in materials subsystem on labour and investment resources

subsystem and their impact on the distribution of labor and investment resources in relevant sectors are plotted in Figs. 3 and 4.

As it can be seen in the Fig. 3, increase in value of funds elasticity in materials subsystem contributes to production of consumer goods. This follows from the fact that (according to increase of elasticity coefficient) efficiency of equipment work in materials subsystem increases and components of workforce in Eq. (8) decreases.

At the same time requirement for workforce in material subsystem decreases. Accordingly, share of allocated investment into this subsystem is also reduced. Also, increase in workforce in means of labor supply subsystem stimulates greater allocation of investment resources into this subsystem.

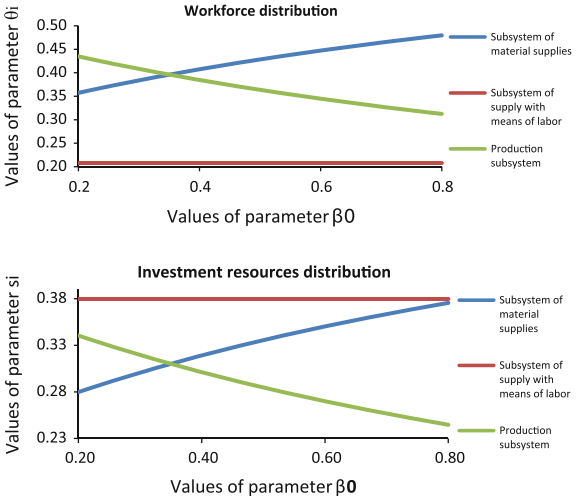

Fig. 4 Effect of changes in material costs in production subsystem on labour and investment resources

According to the Table 3 and Fig. 4, growth of direct material costs of production of the consumer product reduces labor and investment resources in production subsystem, and this leads to decrease in value of production cluster's finished goods.

5 An Optimal Cluster Control Problem

The resulting solution of non-linear programming problem to determine the steady state is an initial step in solving the problem of optimal control with fixed ends of trajectories for economic model of production cluster [13, 14].

Thus, the optimal control of EMPC reduces the problem of dynamic programming when determining of synthesizing controls $\theta_i(k,t)$, $s_i(k,t)$ to maximize cluster production:

$$J(\theta_i, s_i) = \int_0^{\infty} \mathrm{e}^{-\delta t}\theta_2 A_2 k_2^{\alpha_2} dt \to \max \tag{21}$$

under conditions (5), (6) and (12), where δ—discount coefficient.

There is identified a problem of a transition from some initial steady state of a cluster k_i^{init} to the optimal balance state k_i^{end} by redistributing labor—$\theta_i(k,t)$ and investment—$s_i(k,t)$ resources within subsystems that meet the constraints of labor (5), investment (6), material (12) balance and delivering maximum value to discounting of specific consumption (21).

6 Conclusions and Future Works

In this paper, the mathematical model of economical production cluster, given in the form of differential and algebraic equations is built and investigated. In solving the problem there is used Lagrange multipliers of a special kind and developed a new algorithm for finding the optimal steady state of the system. Developed algorithm makes it possible to determine the allocation of capital in the economic model of a production cluster (EMPC).

Also there were investigated the influences of various model parameters on the found optimal steady state of EMPC. To perform calculations according the algorithm MAPLE software package was used. The problem of optimal control with fixed ends of the trajectories to be solved is defined.

Presented methods can be used in an optimal governmental controlled growth of production clusters.

References

1. Porter, M.: On Competition, p. 576. Harvard Business Review Press, Boston (2008)
2. Asherie, N., Chincarini, L.: An analytical model for the formation of economic clusters. Reg. Sci. Urban Econ. **38**(3), 252–270 (2008)
3. Partha, S.: Capital accumulation and convergence in a small open economy. Rev. Int. Econ. **21** (4), 690–704 (2013)
4. Supriyo, D.: Intangible capital and growth in the 'new economy': Implications of a multi-sector endogenous growth model. Struct. Change Econ. Dyn. **28**, 25–42 (2014)
5. Ashimov, A., Sultanov, B., Borovskiy, Y., Ashimov, A., Adilov, Z., Novikov, D.: Macroeconomic analysis and parametric control of a national economy. Springer, US, 288 p (2013)
6. Dumitru, O., Loretti, D., Mihaela, N.: Deterministic and stochastic three-sector dynamic growth model with endogenous labour supply. Econ. Rec. **89**(284), 99–111 (2013)
7. Goncalves, J., Lestas, I., Tonita, R., Vinnicombe, G.: Fluctuations and limitations of a multi-sector economic model with delays. In Proceedings of the Conference on American Control, pp. 6904–6909 (2010)
8. Loveridge, S.: A typology and assessment of multi-sector regional economic impact models. Reg. Stud. **38**(3), 305–317 (2004)
9. Zhang, J.S.: The analytical solution of balanced growth of non-liner dynamic multi-sector economic model. Econ Model. **28**, 410–421 (2011)
10. Ramishvili, I.: Mathematical model of a closed economic system considering delay factor and neutral optimal control problem. In: Several Problems of Applied Mathematics and Mechanics, pp. 145–149 (2013)
11. Kolemayev, V.A.: Economic-mathematical modeling (in Russian), p. 421. UNITY, Moscow (2005)
12. Dasterdi, R.B., Isfahani, R.D.: Equity and economic growth, a theoretical and empirical study: MENA zone. Econ. Model. **28**, 694–700 (2011)
13. Aipanov, Sh., Murzabekov, Z.: Analytical solution of a linear quadratic optimal control problem with control value constraints. Comput. Syst. Sci. Int. **53**(1), 84–91 (2014)
14. Tokarev, V.V.: On the signs of pulses in the problems of optimal control with fixed ends of the trajectories. Autom. Remote Control **62**(8), 1263–1272 (2001)

Verification of the Analytical Traffic Model of a Multidomain IMS/NGN Using the Simulation Model

Sylwester Kaczmarek and Maciej Sac

Abstract In this paper we verify the previously proposed analytical traffic model of a multidomain Next Generation Network (NGN), which is standardized for delivering multimedia services based on the IP Multimedia Subsystem (IMS). For this reason a proper simulation model used, in which not theoretical queuing system models but the operation of real network elements and standardized call scenarios are accurately implemented. Consequently, the simulation model reflects the phenomena taking place in real IMS/NGN network and can be considered as a reference for evaluation of quality of the analytical results. The output variables assessed using both models are mean Call Set-up Delay (CSD) and mean Call Disengagement Delay (CDD), a subset standardized call processing performance parameters. The investigations are performed for various types of calls and sets of network parameters. As a result, conclusions regarding conformity of the analytical and simulation results in a multidomain IMS/NGN are presented.

Keywords IMS · NGN · Call processing performance · Multidomain network · Traffic model · Verification

1 Introduction

The paper concerns the Next Generation Network (NGN) [1], which is an International Telecommunications Union Telecommunication Standardization Sector (ITU-T) proposition for a telecommunication network architecture dedicated to satisfy current and future needs for delivering various multimedia services with guaranteed quality. Providing services in NGN is based on the IP Multimedia

S. Kaczmarek (✉) · M. Sac
Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland
e-mail: kasyl@eti.pg.gda.pl

M. Sac
e-mail: Maciej.Sac@eti.pg.gda.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_9

Subsystem (IMS) concept [2], and therefore the terms "IMS-based NGN" and "IMS/NGN" are very frequently used.

In the paper we present a simulation model a multidomain IMS/NGN, which precisely implements the operation of real network elements and various service scenarios (different variants of registration, intra-operator and inter-operator calls; some calls may be unsuccessful due to transport resource unavailability). The output variables of the model are mean Call Set-up Delay [mean *CSD*, *E(CSD)*] and mean Call Disengagement Delay [mean *CDD*, *E(CDD)*], a subset of standardized call processing performance (CPP) parameters [3, 4], which are very closely related to Quality of Service (QoS). As a result, guaranteeing appropriate values of CPP parameters is crucial for commercial success of IMS/NGN.

In our research the presented simulation model, which reflects the real network operation, is used to verify the previously obtained analytical results based on theoretical queuing system models [5, 6]. The rest of the text is organized as follows. The proposed simulation model of a multidomain IMS/NGN is described in Sect. 2. The results of verification of analytical call processing performance results using the simulation model are presented and discussed in Sect. 3. Summary and future work regarding our traffic models are described in Sect. 4.

2 Simulation Model of a Multidomain IMS/NGN

The multidomain IMS/NGN architecture, which operation is implemented in the simulation model used in this paper, is depicted in Fig. 1. The simulated network consists of two IMS/NGN domains administered by two operators (we use the terms "operator" and "domain" interchangeably with similar meaning). It is assumed that all operators (operator 1 and 2) have their own transport networks with negligible message loss probability, which provide resources necessary for services requested by users. Transport resources of core networks are controlled by RACF C1 and RACF C2 elements, while transport resources of access networks are controlled by RACF A1 and RACF A2 elements. Access network of each operator contains multiple access areas. To perform transport connection between different access areas, resources of the core network are used.

The elements presented in Fig. 1 play the following roles in the network (to mark elements of different operators, numbers 1 and 2 are added to the names listed below) [5, 6]:

- User Equipments (UEs): user terminals generating call set-up and disengagement requests, registering themselves in their domains; it is assumed that UEs perform standard voice calls (application servers are not used) and have compatible codec sets; no audio announcements are played during the calls;
- P-CSCF (Proxy-Call Session Control Function): the server responsible for receiving all messages from UEs and forwarding them to the S-CSCF server;

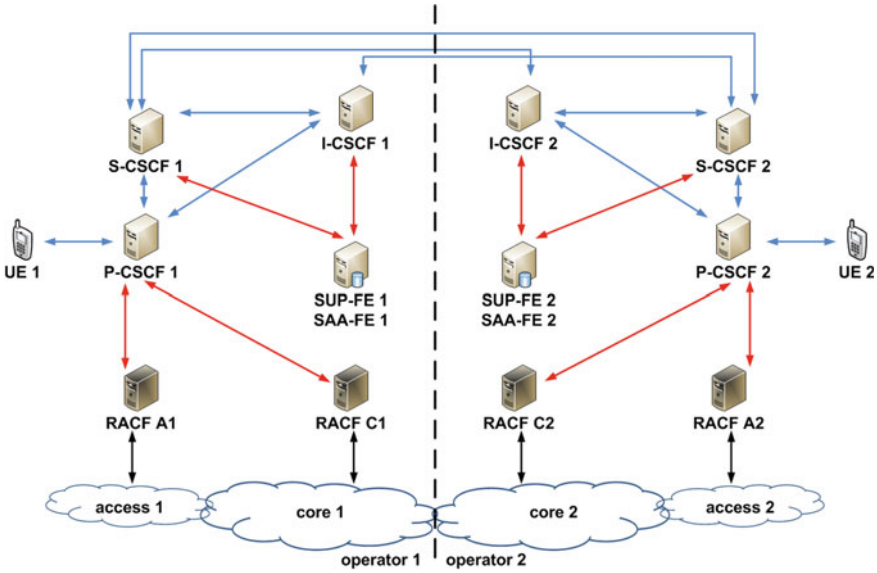

Fig. 1 Model of a multidomain IMS/NGN [5]; *blue arrows*—communication using SIP protocol [11]; *red arrows*—communication using Diameter protocol [12]; *black arrows*—resource control using dedicated protocols

- S-CSCF (Serving-Call Session Control Function): the main server handling all calls in the domain;
- I-CSCF (Interrogating-Call Session Control Function): the server handling messages from other domains;
- SUP-FE/SAA-FE (Service User Profile Functional Entity/Service Authentication and Authorization Functional Entity): the database storing user profiles and location information, performs also authentication and authorization functions;
- RACF (Resource and Admission Control Functions): the unit of the transport stratum allocating resources for a new call and releasing resources during call disengagement; it is assumed that RACF elements control resources using push mode [7]; resources for a call may be unavailable with a defined probability.

The following set of service scenarios is implemented in the model [5, 6]:

- a1—registration of UE (domain 1);
- a2—registration of UE (domain 2);
- b1—intra-operator call (domain 1, the same access area, success);
- b2—intra-operator call (domain 2, the same access area, success);
- c1—intra-operator call (domain 1, the same access area, resources unavailable);
- c2—intra-operator call (domain 2, the same access area, resources unavailable);

- d1—intra-operator call (domain 1, different access areas, success);
- d2—intra-operator call (domain 2, different access areas, success);
- e1—intra-operator call (domain 1, different access areas, resources unavailable);
- e2—intra-operator call (domain 2, different access areas, resources unavailable);
- f1—inter-operator call (originated in domain 1, success);
- f2—inter-operator call (originated in domain 2, success);
- g1—inter-operator call (originated in domain 1, resources unavailable only in originating domain or in both domains);
- g2—inter-operator call (originated in domain 2, resources unavailable only in originating domain or in both domains);
- h1—inter-operator call (originated in domain 1, resources unavailable only in terminating domain);
- h2—inter-operator call (originated in domain 2, resources unavailable only in terminating domain).

Due to space limitations, service scenarios are not described in this paper. For more details about the scenarios please refer to [5–7, 8, 9].

Structure of the simulation model of a multidomain IMS/NGN is presented in Fig. 2. For implementation of the simulator the OMNeT++ framework [10] was chosen due to its flexibility, very good documentation, possibility of applying standard C/C++ libraries, as well as our previous experience. In the OMNeT++ framework simulated objects (called modules), their parameters and connections between them are defined using NED (NEtwork Description) language [10]. Basic operations in the developed application are performed by simple modules, which functionality is described using C++ programming language. Simple modules may be grouped to form compound modules, which perform more complicated functions.

In the model six compound modules were implemented: UE, PCSCF, SCSCF, ICSCF, RACF, SUPFE_SAAFE (Fig. 2), which perform the role of the UE, P-CSCF, S-CSCF, I-CSCF, RACF, SUP-FE/SAA-FE elements respectively. Our aim was to make the implemented modules as universal as possible—their instances can be assigned to domain 1 or 2. For the RACF module controlled transport network type (access, core) is additionally definable. As a result, using the above mentioned six compound modules complete IMS/NGN architecture depicted in Fig. 1 was created.

Each implemented compound module includes the "*_main" simple module performing operations specific for the compound module and "l2tran" simple modules (one for each neighbor; Fig. 2 at the bottom). The "l2tran" modules are responsible for buffering messages in FIFO queues before sending them through channels, which represent communication links. This operation is not included in the channels provided by the OMNeT++ and must be implemented separately. It is worth mentioning that messages coming to each compound module are not buffered and are delivered directly to the "*_main" simple module.

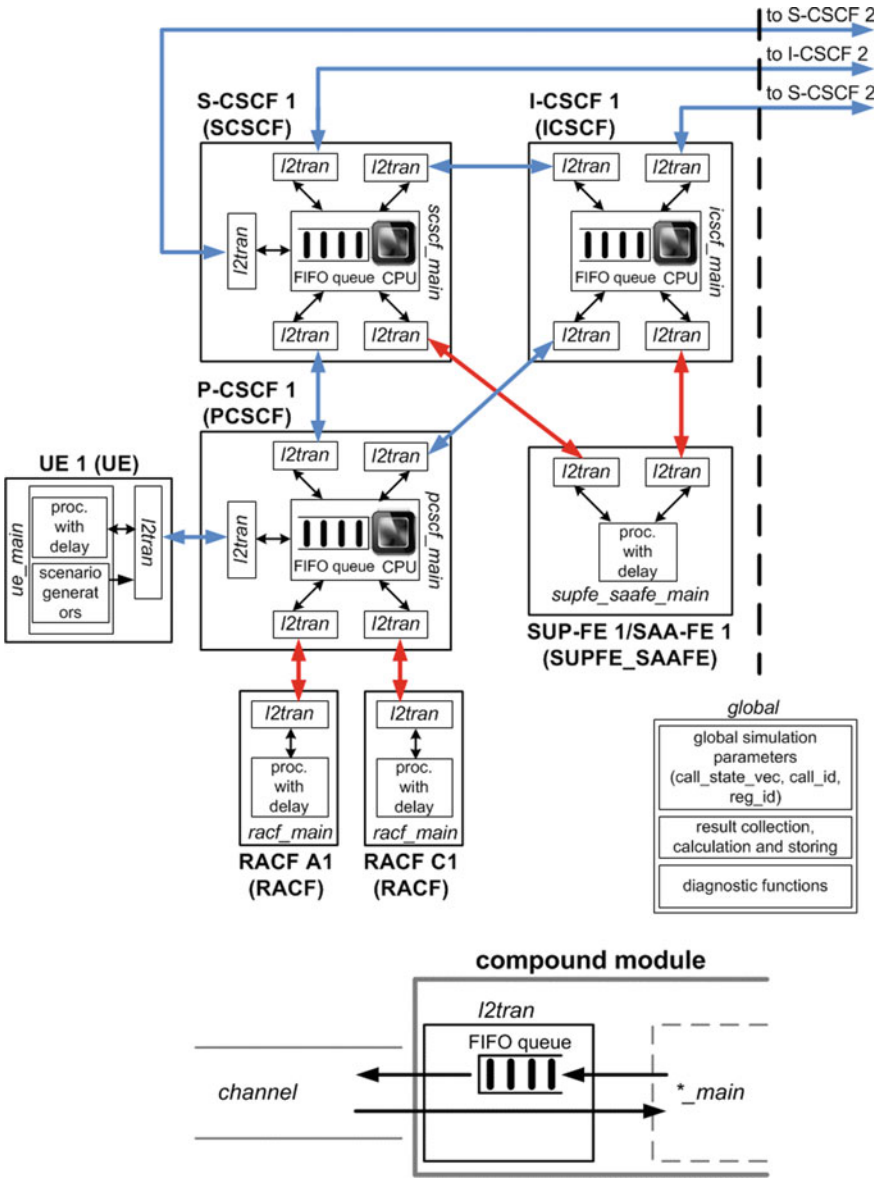

Fig. 2 Structure of the simulation model (*top*) and of the l2tran simple module (*bottom*); names of compound modules are written using *bold font*, *italic font* represents names of simple modules, functionality of particular modules is depicted using *normal font*; for better clarity only structure of domain 1 is presented, structure of domain 2 is its mirror image

In PCSCF, SCSCF and ICSCF compound modules "*_main" simple modules store messages in FIFO queues and handle them using Central Processing Units (CPUs) according to the service scenario (message processing time depends on the message and scenario type). As a result of processing, the message can be forwarded to another module or a new message can be generated. In order to be correctly identified and processed, messages used in the simulator contain standardized SIP [11] and Diameter [12] protocol fields, such as Via, From, To (for SIP protocol) and Call-Id (for both SIP and Diameter protocol). These fields are analyzed and processed by all modules. The most complex of the *CSCF modules is the PCSCF module, which handles responses regarding allocation and release of transport resources from RACF modules.

As the UE compound module represent many terminals simultaneously performing service scenarios, it does not contain a queue in the "ue_main" simple module. The aim of the "ue_main" simple module is to respond for each received message after a specified time (to process message with a particular delay—the "proc. with delay" block in Fig. 2), which depends on the received message type. Another very important task of the UE compound module is service request generation. For aggregated registration requests as well as intra- and inter-domain call requests exponential intervals are assumed with definable intensities. The UE module is also responsible for introducing delays corresponding to user behavior, such as call answer delay and call duration [13]. These delays are definable simulation parameters.

The RACF and SUPFE_SAAFE modules process messages similarly to the UE module (they contain "proc. with delay" blocks), however, in these modules message processing time does not depend on the type of the incoming message (it is given by a random variable for all messages). The RACF compound module can give a negative response for resource allocation request with a definable probability.

The last module, which still has not been described, is the "global" simple module. There is only one instance of this module and it is crucial for the simulator due to its several tasks. The first task of the "global" module is to provide the other simulated modules with global simulation variables (for example with identifiers of registration and call requests, which must be unique) via its global functions. Global functions the "global" module are also used for diagnostic purposes, such as displaying full content of messages. The "global" module also stores a vector with structures describing states of all generated calls named the call_state_vec. Elements of this vector are accessed and updated by all simulated modules. The updated fields include among other the t_1–t_{10} times, which are used to calculate *CSD* and *CDD* for a single call based on the definition given by ITU-T [3, 4]. *CSD* and *CDD* values gathered for all calls are used to obtain the output variables of the simulation model, which are mean Call Set-up Delay and mean Call Disengagement Delay as well as associated confidence intervals for defined confidence levels. *E(CSD)* and *E(CDD)* are computed for all successful call scenarios (b1, b2, d1, d2, f1, f2). Indexes are added to indicate the scenario name.

For example, $E(CSD)_{f1}$ and $E(CDD)_{f1}$ regard the f1 scenario. Final simulation results along with detailed network configuration are written to a result text file, which default name is result.txt.

To operate properly, the simulation model of a multidomain IMS/NGN requires values of several input variables, from which the most important are:

- λ_{R1}, λ_{R2}: registration request (SIP REGISTER) intensity in domain 1 and 2 respectively;
- λ_{11}, λ_{22}: intra-operator call set-up request (SIP INVITE) intensity (domain 1 and 2 respectively);
- λ_{12}, λ_{21}: inter-operator call set-up request (SIP INVITE) intensity (requests originated in domain 1 and 2 respectively);
- r_{C1}, r_{C2}: ratio of calls involving multiple access areas to all intra-operator calls (domain 1 and 2 respectively);
- p_{A1}, p_{C1}, p_{A2}, p_{C2}: probability of transport resource unavailability in access 1, core 1, access 2 and core 2 (Fig. 1) respectively;
- T_{INV_P1}, T_{INV_S1}, T_{INV_I1}, T_{INV_P2}, T_{INV_S2}, T_{INV_I2}: times of processing the SIP INVITE message by P-CSCF 1, S-CSCF 1, I-CSCF 1, P-CSCF 2, S-CSCF 2, I-CSCF 2 respectively;
- a_k ($k = 1, 2, \ldots, 24$): the factors determining times of processing other SIP and Diameter messages by CSCF servers CPUs as follows: $T_{TR\_*} = a_1 \cdot T_{INV\_*}$, $T_{RING\_*} = a_2 \cdot T_{INV\_*}$, $T_{OKINV\_*} = a_3 \cdot T_{INV\_*}$, $T_{ACK\_*} = a_4 \cdot T_{INV\_*}$, $T_{BYE\_*} = a_5 \cdot T_{INV\_*}$, $T_{OKBYE\_*} = a_6 \cdot T_{INV\_*}$, $T_{AAA\_*} = a_7 \cdot T_{INV\_*}$, $T_{STA\_*} = a_8 \cdot T_{INV\_*}$, $T_{PRA\_*} = a_9 \cdot T_{INV\_*}$, $T_{UPD\_*} = a_{10} \cdot T_{INV\_*}$, $T_{SP\_*} = a_{11} \cdot T_{INV\_*}$, $T_{OKUPD\_*} = a_{12} \cdot T_{INV\_*}$, $T_{OKPRA\_*} = a_{13} \cdot T_{INV\_*}$, $T_{LIA\_*} = a_{14} \cdot T_{INV\_*}$, $T_{SERUN\_*} = a_{15} \cdot T_{INV\_*}$, $T_{CAN\_*} = a_{16} \cdot T_{INV\_*}$, $T_{OKCAN\_*} = a_{17} \cdot T_{INV\_*}$, $T_{PRFAIL\_*} = a_{18} \cdot T_{INV\_*}$, $T_{REG\_*} = a_{19} \cdot T_{INV\_*}$, $T_{UAA\_*} = a_{20} \cdot T_{INV\_*}$, $T_{MAA\_*} = a_{21} \cdot T_{INV\_*}$, $T_{UNAUTH\_*} = a_{22} \cdot T_{INV\_*}$, $T_{SAA\_*} = a_{23} \cdot T_{INV\_*}$, $T_{OKREG\_*} = a_{24} \cdot T_{INV\_*}$; where "*" can be replaced by "P1", "S1", "I1", "P2", "S2", "I2", which represent P-CSCF 1, S-CSCF 1, I-CSCF 1, P-CSCF 2, S-CSCF 2, I-CSCF 2 respectively; the a_k factors concern the following messages: 100 Trying, 180 Ringing, 200 OK (INVITE), ACK, BYE, 200 OK (BYE), AAA, STA, PRACK, UPDATE, 183 Session Progress, 200 OK (UPDATE), 200 OK (PRACK), LIA, 503 Service Unavailable, CANCEL, 200 OK (CANCEL), 580 Precondition Failure, REGISTER, UAA, MAA, 401 Unauthorized, SAA, 200 OK (REGISTER);
- X_{A1}, X_{C1}, X_{A2}, X_{C2}, Y_1, Y_2: message processing times by RACF A1, RACF C1, RACF A2, RACF C2, SUP-FE 1/SAA-FE 1, SUP-FE 2/SAA-FE 2 respectively (random variables);
- lengths and bandwidths of optical links, lengths of transmitted messages: values determining communication times between network elements.

Due to limited space, more details about the implementation of the simulator cannot be provided in this paper. Some additional information on this matter can be

found in paper [13] dedicated to the simulation model of a single IMS/NGN domain, based on which the multidomain model was developed. It is, however, very important that some significant modifications were made to the code described in [13], so certain details presented in paper [13] may be outdated.

Correctness of the proposed simulation model was verified for many runs with different input variables by thorough analysis of OMNeT++ event log (containing information about all events occurring during simulation) using the Sequence Chart tool [10]. The investigations indicated that operation of all elements and all service scenarios is properly performed.

Due to the fact that in the described simulation model the operation of real network elements and standardized call scenarios are accurately implemented (the simulated elements process SIP and Diameter messages according to standards), it reflects the phenomena taking place in real IMS/NGN network. Therefore, it can be considered as a reference for evaluation of quality of the analytical results. In the analytical model mean values of theoretical delays introduced by the elements depicted in Figs. 1 and 2 are calculated and appropriately summed to obtain mean *CSD* and mean *CDD* [5, 6]. These component delays include:

- mean message waiting times in communication queues (implemented in the simulation model as l2tran modules; Fig. 2); for calculation of these times mathematical model of M/G/1 queuing system is used;
- message transmission times (message lengths divided by links bandwidths),
- propagation times (equal to 5 μs/km—optical links are assumed);
- mean message waiting times in CSCF servers CPU queues (Fig. 2), which store incoming messages when CSCF servers CPUs are busy (for calculation of these times mathematical model of M/G/1 queuing system is used);
- mean message processing times by CSCF servers CPUs as well as RACF and SUP-FE/SAA-FE elements (Fig. 2); for CSCF servers message processing times are calculated using the T_{INV_P1}, T_{INV_S1}, T_{INV_I1}, T_{INV_P2}, T_{INV_S2}, T_{INV_I2}, a_k input variables; for RACF and SUP-FE/SAA-FE elements mean values of the X_{A1}, X_{C1}, X_{A2}, X_{C2}, Y_1, Y_2 random variables are used in calculations.

3 Results

The aim of the investigations presented in this paper was to verify the analytical call processing performance results in a multidomain IMS/NGN [5, 6] using the simulation model described in Sect. 2. The relations between particular input variables (Sect. 2) and call processing performance of IMS/NGN were studied in [5, 6] and are out of scope of this paper. The examined input data sets are presented in Tables 1 and 2 correspondingly.

Table 1 Input data sets selected for verification of the analytical results presented in [5]

Anal. data set	λ_R (1/s)	λ_{1d} (1/s)	λ_{2d} (1/s)	r_C	p_b	T_{INV} (ms)	X (ms)	Y (ms)	d (km)	b (Mb/s)	m_1	m_2
[5], 1	1–127.5	110	50	0.5	0	0.5	10	10	200	50	–	–
	1–279	80	50	0.5	0	0.5	10	10	200	50	–	–
	350	1–65.8	50	0.5	0	0.5	10	10	200	50	–	–
	200	1–95.8	50	0.5	0	0.5	10	10	200	50	–	–
[5], 2	50	50	1–56.5	0.5	0	0.7	10	10	200	50	–	–
	50	50	1–72.2	0.5	0	0.6	10	10	200	50	–	–
	50	50	90	0.5	0	0.1–0.52	10	10	200	50	–	–
	50	50	61	0.5	0	0.1–0.67	10	10	200	50	–	–
[5], 3	50	50	50	0.01, 0.5, 0.99	0, 0.05, 0.1	0.5	10	10	200	50	–	–
[5], 4	50	50	50	0.5	0	0.5	5, 200, 400	5, 200, 400	200	50	–	–
[5], 5	50	50	50	0.5	0	0.5	10	10	1, 500, 1000	5.6–200	–	–

When there are more than three vales of particular input variable, only a range of the variable is indicated; the asterisk symbol (*) represents a wildcard pattern (one or more characters of any type)

Table 2 Input data sets selected for verification of the analytical results presented in [6]

Anal. data set	λ_R (1/s)	λ_{1d} (1/s)	λ_{2d} (1/s)	r_C	p_b	T_{INV} (ms)	X (ms)	Y (ms)	d (km)	b (Mb/s)	m_1	m_2
[6], 1	$\lambda_{R1} = 50$ $\lambda_{R2} = m_1 \cdot \lambda_{R1}$	$\lambda_{11} = 50$ $\lambda_{22} = m_2 \cdot \lambda_{11}$	50	0.5	0	0.5	10	10	200	50	0.01–1.17	2.5
			50	0.5	0	0.5	10	10	200	50	0.01–3.7	2
			50	0.5	0	0.5	10	10	200	50	5	0.01–1.72
			50	0.5	0	0.5	10	10	200	50	3	0.01–2.13
[6], 2	50	50	$\lambda_{12} = 50$ $\lambda_{21} = m_1 \cdot \lambda_{12}$	0.5	0	$T_{INV\_*1} = 0.5$ $T_{INV\_*2} = m_2 \cdot T_{INV\_*1}$	10	10	200	50	0.1–2.885	1.6
	50	50		0.5	0		10	10	200	50	0.1–2.885	0.75
	50	50		0.5	0		10	10	200	50	2.5	0.01–1.05
	50	50		0.5	0		10	10	200	50	1.5	0.01–1.33
[6], 3	50	50	50	$r_{C1} = 0.5$ $r_{C2} = 0.01$, 0.5, 0.99	$p_{*1} = 0$ $p_{*2} = 0$, 0.05, 0.1	0.5	10	10	200	50	–	–
[6], 4	50	50	50	0.5	0	0.5	$X_{*1} = 10$ $X_{*2} = m_1 \cdot X_{*1}$	$Y_1 = 10$ $Y_2 = m_2 \cdot Y_1$	200	50	0.01, 10, 20	0.01, 10, 20
[6], 5	50	50	50	0.5	0	0.5	10	10	$d_1 = 200$ $d_2 = m_1 \cdot d_1$	$b_1 = 50$ $b_2 = m_2 \cdot b_1$	0.005, 2.5, 5	0.112–2

When there are more than three vales of particular input variable, only a range of the variable is indicated; the asterisk symbol (*) represents a wildcard pattern (one or more characters of any type)

In these tables the following notations are used:

- $\lambda_{R1} = \lambda_{R2} = \lambda_R$, $\lambda_{11} = \lambda_{22} = \lambda_{1d}$, $\lambda_{12} = \lambda_{21} = \lambda_{2d}$, $r_{C1} = r_{C2} = r_C$, $p_{A1} = p_{C1} = p_{A2} = p_{C2} = p_b$;
- $T_{INV\_P1} = T_{INV\_S1} = T_{INV\_I1} = T_{INV\_P2} = T_{INV\_S2} = T_{INV\_I2} = T_{INV}$;
- $X_{A1} = X_{C1} = X_{A2} = X_{C2} = X$, $Y_1 = Y_2 = Y$;
- d_1, d_2: lengths of links in domain 1 and 2, $d_1 = d_2 = d$;
- b_1, b_2: bandwidths of links in domain 1 and 2, $b_1 = b_2 = b$.

Apart from Tables 1 and 2, the following values of the remaining input variables were used:

- $\{a_1, \ldots, a_{24}\}$ = {0.2, 0.2, 0.6, 0.3, 0.6, 0.3, 0.6, 0.6, 0.3, 0.5, 0.6, 0.5, 0.3, 0.5, 0.4, 0.6, 0.4, 0.4, 0.6, 0.5, 0.5, 0.3, 0.4, 0.3};
- SIP message lengths in bytes [14]: *INVITE*: 930, *PRACK*: 450, *UPDATE*: 930, *ACK*: 630, *BYE*: 510, *CANCEL*: 550, *REGISTER*: 700, *100 Trying*: 450, *180 Ringing*: 450, *183 Session Progress*: 910, *200 OK (INVITE, UPDATE)*: 990, *200 OK (PRACK, BYE, REGISTER, CANCEL)*: 500, *401 Unauthorized*: 700, *503 Service Unavailable*: 500, *580 Precondition Failure*: 550;
- Diameter message length: 750 bytes.

For the data sets presented in Table 1 parameters of the elements and traffic levels in both IMS/NGN domains are the same (we call this situation the symmetric case). This results in the same values of call processing performance parameters for inter-operator calls originated in domain 1 and 2 ($E(CSD)_{f1} = E(CSD)_{f2} = E(CSD)_{2d}$ and $E(CDD)_{f1} = E(CDD)_{f2} = E(CDD)_{2d}$). An analogical situation takes place for intra-operator calls ($E(CSD)_{b1} = E(CSD)_{b2} = E(CSD)_{d1} = E(CSD)_{d2} = E(CSD)_{1d}$ and $E(CDD)_{b1} = E(CDD)_{b2} = E(CDD)_{d1} = E(CDD)_{d2} = E(CDD)_{1d}$). The same results for b1, d1 as well as b2, d2 scenarios are caused by the fact that mean message processing times by all RACF elements are identical [5].

Using the data sets presented in Table 2 we investigate asymmetric cases, in which values of two selected input variables in domain 2 are modified (multiplied by the m_1 and m_2 factors), while the corresponding values in domain 1 are unchanged. Similarly to the symmetric case, mean message processing times by RACF elements in particular domains are the same (but generally not the same in the whole network). Therefore the results for b1, d1 as well as b2, d2 scenarios are identical [6]:

- $E(CSD)_{b1} = E(CSD)_{d1} = E(CSD)_{b1/d1}$;
- $E(CSD)_{b2} = E(CSD)_{d2} = E(CSD)_{b2/d2}$;
- $E(CDD)_{b1} = E(CDD)_{d1} = E(CDD)_{b1/d1}$;
- $E(CDD)_{b2} = E(CDD)_{d2} = E(CDD)_{b2/d2}$.

Apart from call processing performance parameters for intra-domain calls, in the asymmetric case also performance of inter-domain calls is examined ($E(CSD)_{f1}$, $E(CSD)_{f2}$, $E(CDD)_{f1}$, $E(CDD)_{f2}$).

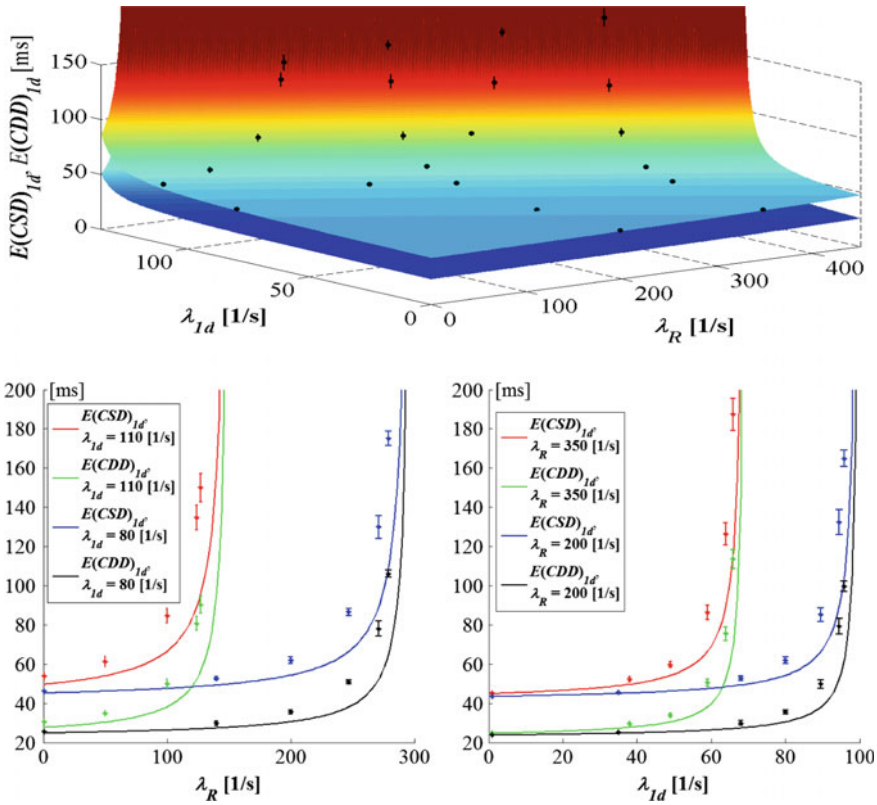

Fig. 3 Verification of the analytical results for data set 1 from [5] (Table 1)—intra-domain calls (scenarios b1/d1 and b2/d2)

The results of selected investigations (Tables 1 and 2) are presented in Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12. Due to space limitations, all obtained results cannot be included in the paper, however, the presented conclusions are drawn from all performed experiments. Each presented figure consists of three subfigures. Subfigures at the top depict the analytical results obtained for the data sets from [5, 6]—two surfaces representing mean *CSD* and mean *CDD* for a particular type of call scenario. Due to the fact that for all scenarios call set-up process is more complicated than call disengagement (more messages are sent and processed), the surfaces for mean *CSD* are always above these for mean *CDD*.

Subfigures at the top also include the simulation results (points with confidence intervals), which are used to verify the analytical results. Due to longer duration of simulations, input data sets used in simulations are subsets of these used in calculations (Tables 1 and 2). Our aim during selection of the simulation data sets was to provide more simulation results in areas of significant changes in mean *CSD* and *CDD*.

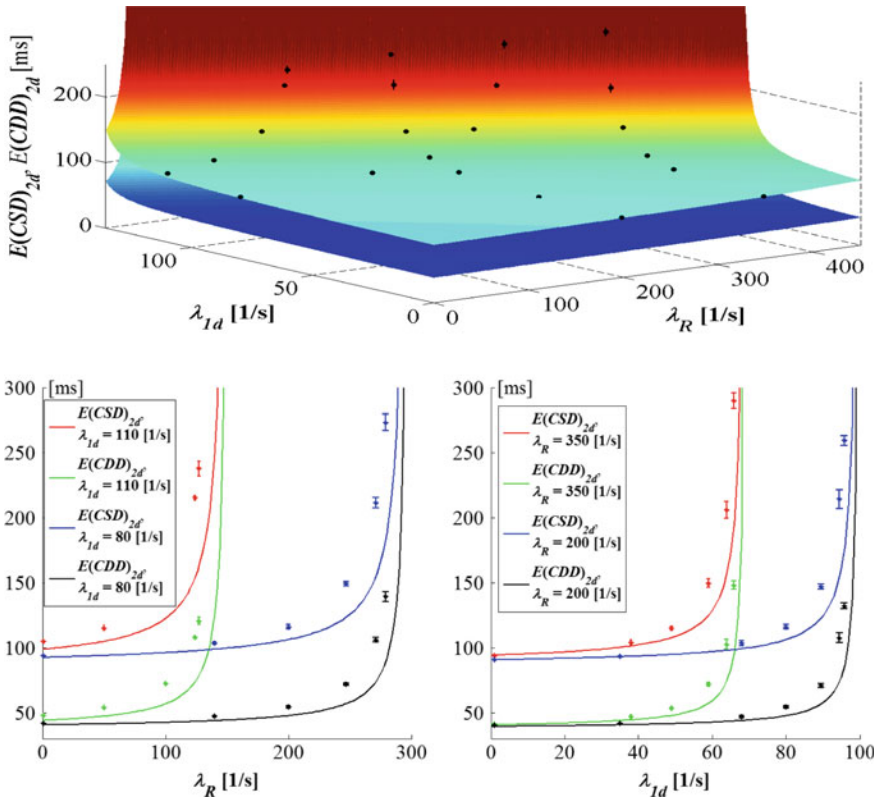

Fig. 4 Verification of the analytical results for data set 1 from [5] (Table 1)—inter-domain calls (scenarios f1 and f2)

The simulation results presented in the top subfigures should be considered as an illustration of selection of the simulation data sets (for better clarity only the simulation results for mean *CSD* are marked). The comparisons of the simulation and analytical results (for both mean *CSD* and *CDD*) are provided in the subfigures at the bottom. In these subfigures analytical results are marked using solid lines and simulation results are represented by points with confidence intervals.

Simulations for all data sets presented in Tables 1 and 2 were performed with the following parameters:

– warm-up period: 1250 s;
– 5 measurement periods;
– 0.95 confidence level;

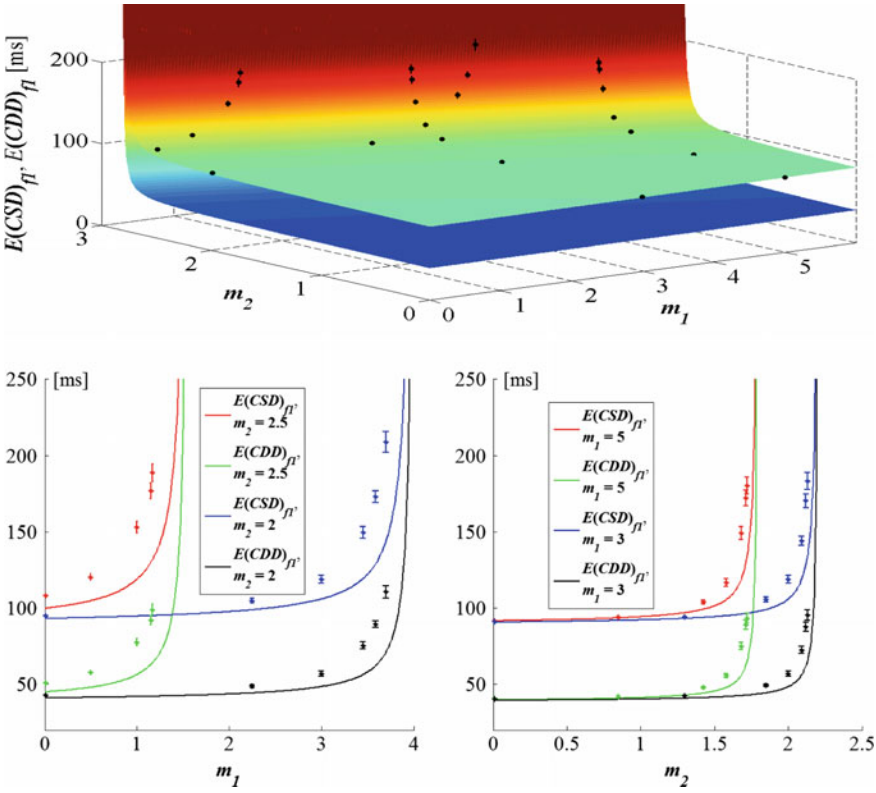

Fig. 5 Verification of the analytical results for data set 1 from [6] (Table 2)—inter-domain calls (scenario f1)

and stop conditions:

- for each type of generated call scenarios there are at least 5000 finished calls and confidence intervals for mean *CSD* as well as *CDD* do not exceed 5 % of mean Call Set-up Delay and mean Call Disengagement Delay;
- total number of generated calls exceeds 10,000,000;
- total simulation time exceeds 10 h.

The performed research indicated that in the majority of cases analytical call processing performance results obtained using M/G/1 queuing system models underestimate the simulation results, which are considered as a reference (Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12). This fact is especially visible for higher loads of CSCF servers, which conforms to the observations from [7] obtained using the simulation model of a single IMS/NGN domain.

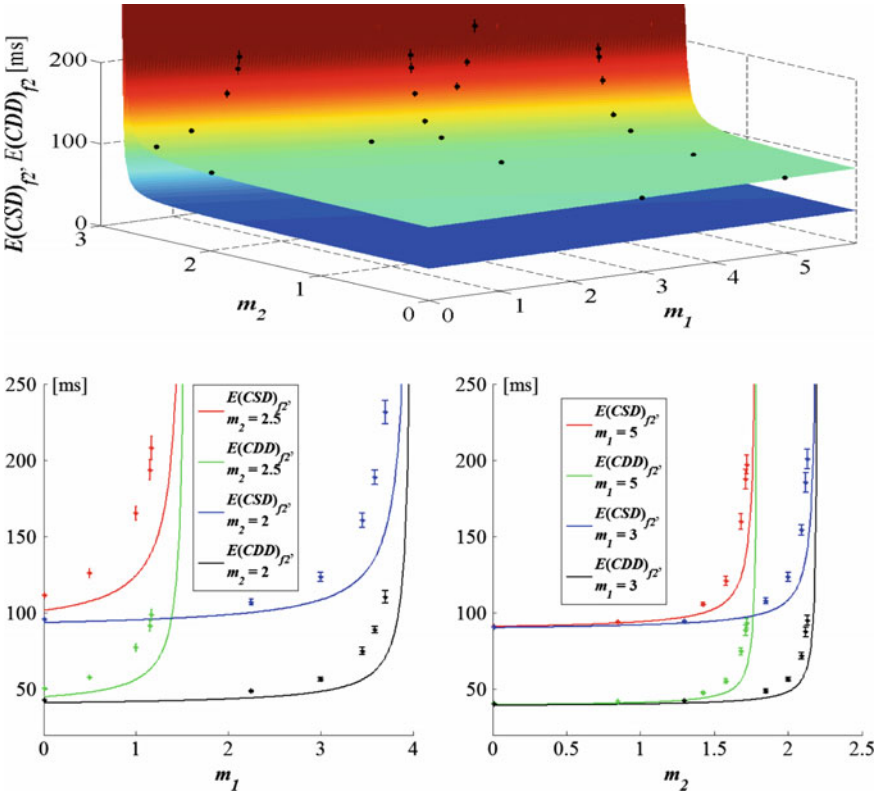

Fig. 6 Verification of the analytical results for data set 1 from [6] (Table 2)—inter-domain calls (scenario f2)

Only in the situation when communication links are overloaded (CSCF servers work normally) there are some simulation results, which are very slightly below the calculations (data sets [5], 5 and [6], 5; due to space limitations these situations are not visualized in the paper). The difference between calculations and simulations in such cases is almost invisible without zooming, contrary to the situation from [7], where it is more easily observable.

As the obtained differences between simulations and calculations in a multidomain IMS/NGN are in most cases similar to these described in [7], we believe that the approach to increase the conformity of analytical and simulation results applied in a single IMS/NGN domain will be also appropriate in a multidomain environment. Therefore, in the near future we are going to replace M/G/1 queuing system models in the analytical model of a multidomain IMS/NGN, with PH/PH/1 queuing

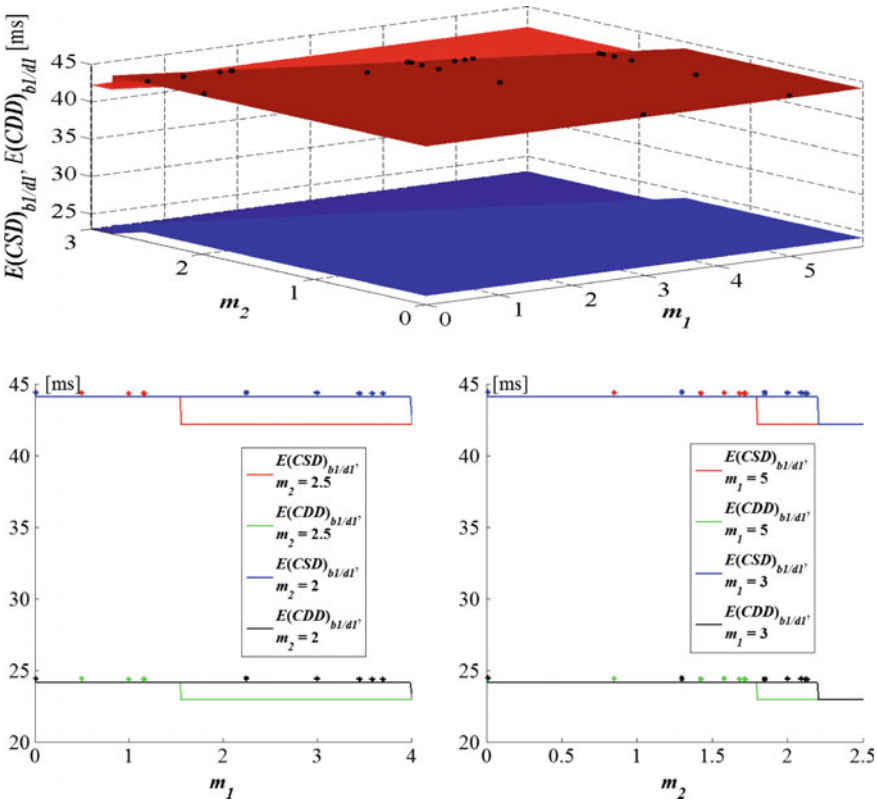

Fig. 7 Verification of the analytical results for data set 1 from [6] (Table 2)—intra-domain calls (scenario b1/d1)

system models [15, 16], which offered very good results in our previous research regarding a single domain of IMS/NGN [15, 16].

Apart from the previously mentioned aspects, the experiments performed using the simulation model allowed some improvements in the analytical model. The applied improvements concerned the situation when one of IMS/NGN domains is overloaded. In such a case intra-domain calls can be normally performed in the other domain. Moreover, mean *CSD* and mean *CDD* for these calls are lower. This situation is caused by the fact that no call set-up requests from the overloaded domain are sent to the other domain and every call set-up request that is sent to the overloaded domain is lost. As a result, the load of the normally operating domain is lower.

The above mentioned situation is covered by data sets [6], 1 and [6], 2 from Table 2. We can observe that when registration intensity ($\lambda_{R2} = m_1 \cdot \lambda_{R1}$) or intra-domain call set-up intensity ($\lambda_{22} = m_2 \cdot \lambda_{11}$) causes domain 2 to be

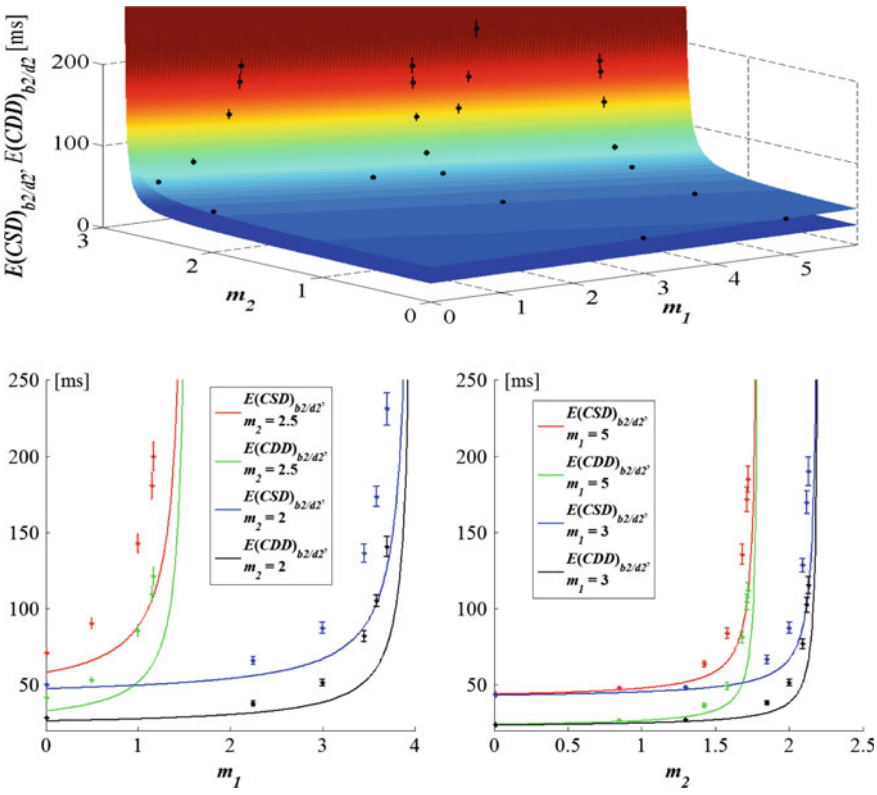

Fig. 8 Verification of the analytical results for data set 1 from [6] (Table 2)—intra-domain calls (scenario b2/d2)

overloaded (very high or infinite values of mean *CSD* and mean *CDD* in Figs. 5, 6 and 8), mean *CSD* and mean *CDD* for intra-domain calls in domain 1 decreases (Fig. 7). Similarly, inter-domain call set-up request intensity in domain 2 ($\lambda_{21} = m_1 \cdot \lambda_{12}$) affects mean *CSD* and mean *CDD* for intra-domain calls generated in domain 1 (Fig. 11) as long as domain 2 is not overloaded due to high $T_{INV\_*2} = m_2 \cdot T_{INV\_*1}$. Otherwise, inter-domain call set-up requests from domain 2 do not reach domain 1, which results in lower mean CSD and mean CDD for intra-domain calls generated in domain 1 (Fig. 11).

The described above operation of a multidomain IMS/NGN was not included in the previous version of the analytical model used in [5, 6]. Therefore former results for data sets [6], 1 and [6], 2 (Figs. 3 and 5 in [6]) differ from the latest results obtained using the updated analytical model and presented in this paper (Figs. 7 and 11).

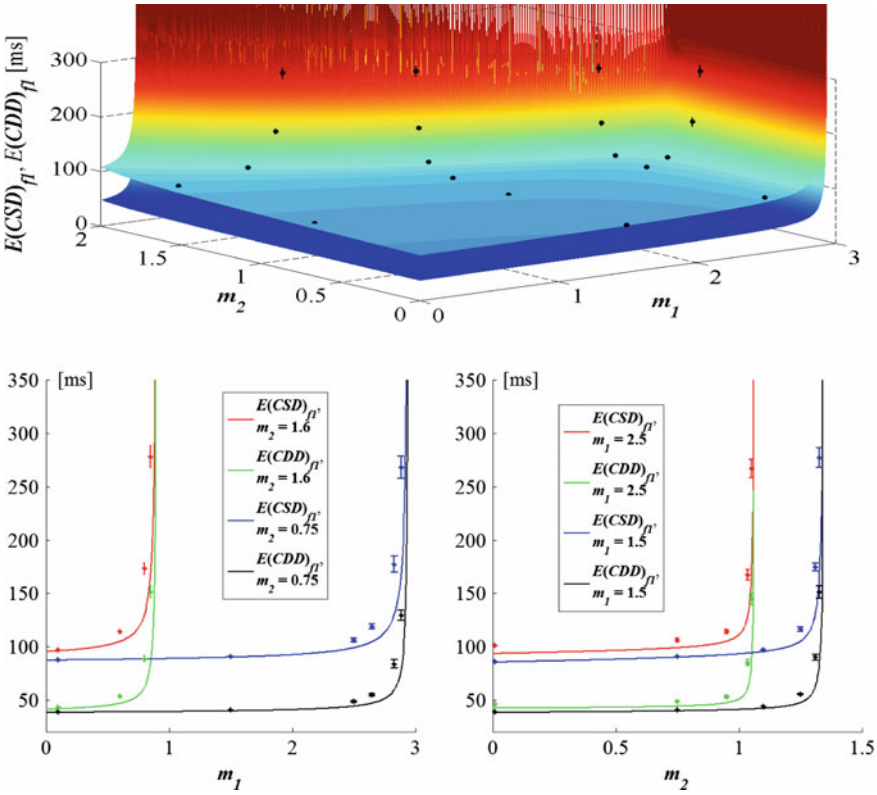

Fig. 9 Verification of the analytical results for data set 2 from [6] (Table 2)—inter-domain calls (scenario f1)

4 Conclusions and Future Work

The paper is a continuation of our research concerning call processing performance in a multidomain IMS/NGN controlled by two network operators. Our aim is to verify the previously performed analytical results based on theoretical M/G/1 queuing system models [5, 6] using the simulation model, which reflects the phenomena occurring in real IMS/NGN network. Both the simulation model and the results of the verification are presented in this paper.

The simulation model was implemented using the OMNeT++ framework and takes into account an extensive set of network parameters and service scenarios (e.g. terminal registration as well as intra-operator and inter-operator calls, which can be successful or unsuccessful due to transport resource unavailability). The output variables of the model are mean Call Set-up Delay and mean Call Disengagement Delay (a set of call processing performance parameters defined by

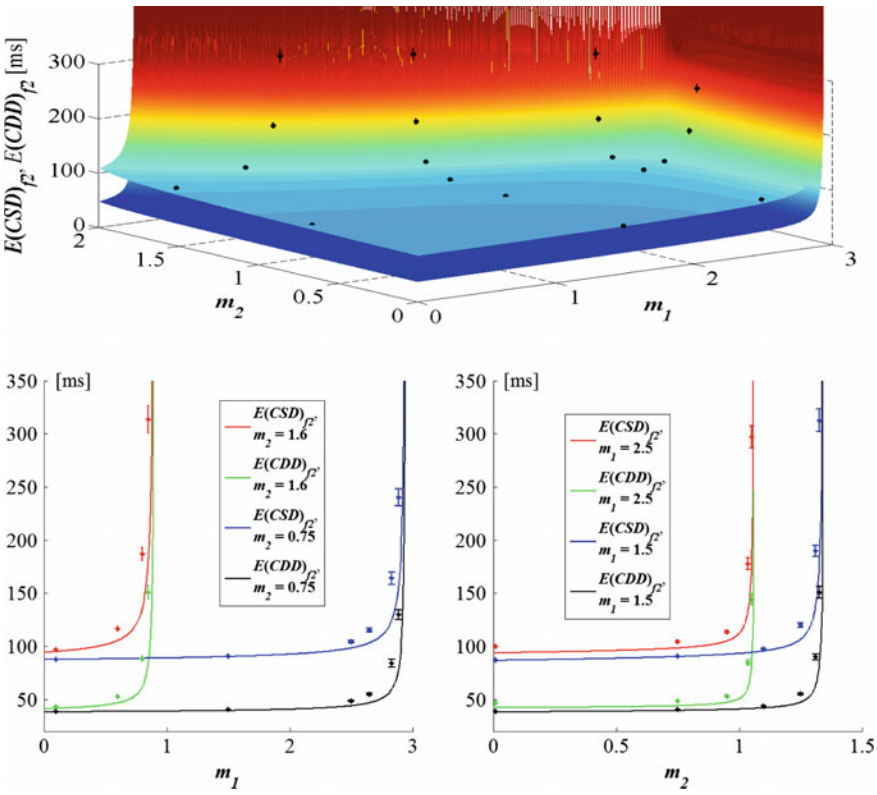

Fig. 10 Verification of the analytical results for data set 2 from [6] (Table 2)—inter-domain calls (scenario f2)

ITU-T) along with corresponding confidence intervals for both intra-operator and inter-operator calls.

The advantage of the presented model is a modular design with core functionalities performed by simple modules forming compound modules. The implemented compound modules are universal—they can be simply configured to work in both domains. Additionally, the RACF module can be configured to control either access or core network. As a result of such an approach, the complete IMS/NGN architecture simulated in this paper is based only on six compound modules. The modular design also allows easy extensions of the simulation model by adding new functionalities or network elements, which can be done by modifications of existing modules or introduction of new modules.

The described simulation model was used to verify the previously performed analytical results presented in [5, 6]. The verification indicated that in the majority of cases the analytical results obtained using M/G/1 queuing system models are lower than the reference simulation results. These discrepancies are quite slight for lower system loads and in such conditions M/G/1 queuing system models can lead

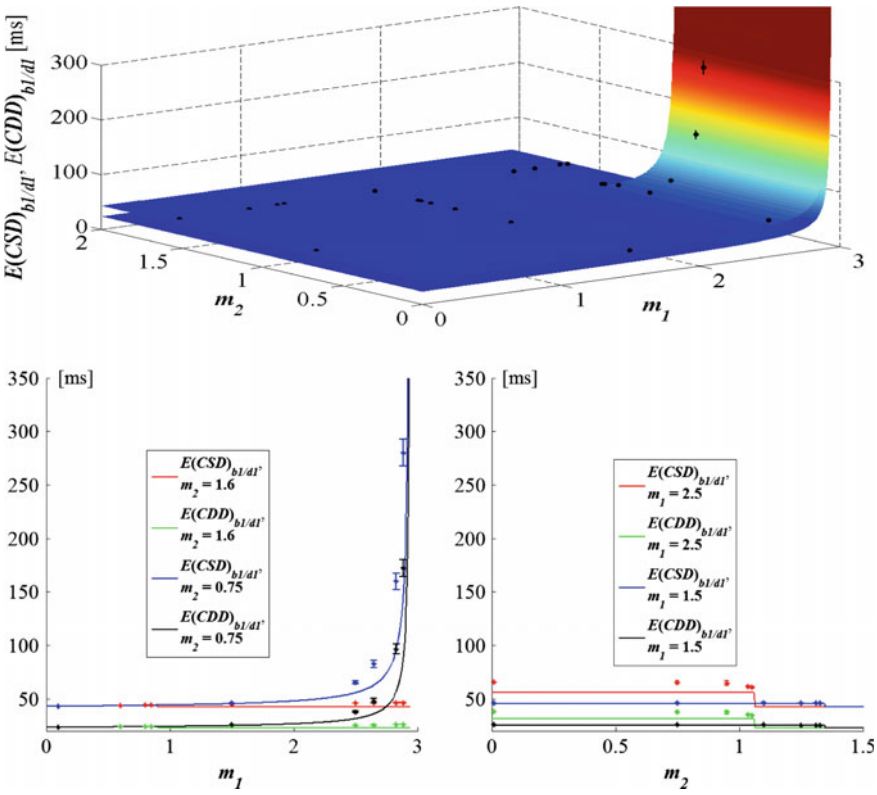

Fig. 11 Verification of the analytical results for data set 2 from [6] (Table 2)—intra-domain calls (scenario b1/d1)

to satisfactory results. However, for higher loads differences between simulations and calculations are significant and it seems that other queuing system models should be investigated. To solve this issue we are going to apply PH/PH/1 queuing system models in our analytical model, which in our previous research in a single IMS/NGN domain [7] successfully improved the conformity of calculations and simulations. We are going to investigate the above mentioned aspects in the next step of our research.

The investigations performed using the simulation model also allowed some improvements in the analytical model, which did not take into consideration the situation when one of IMS/NGN domains is overloaded and the other one works normally. Taking this aspect into account, the analytical model was revised and updated analytical results are presented and verified in this paper.

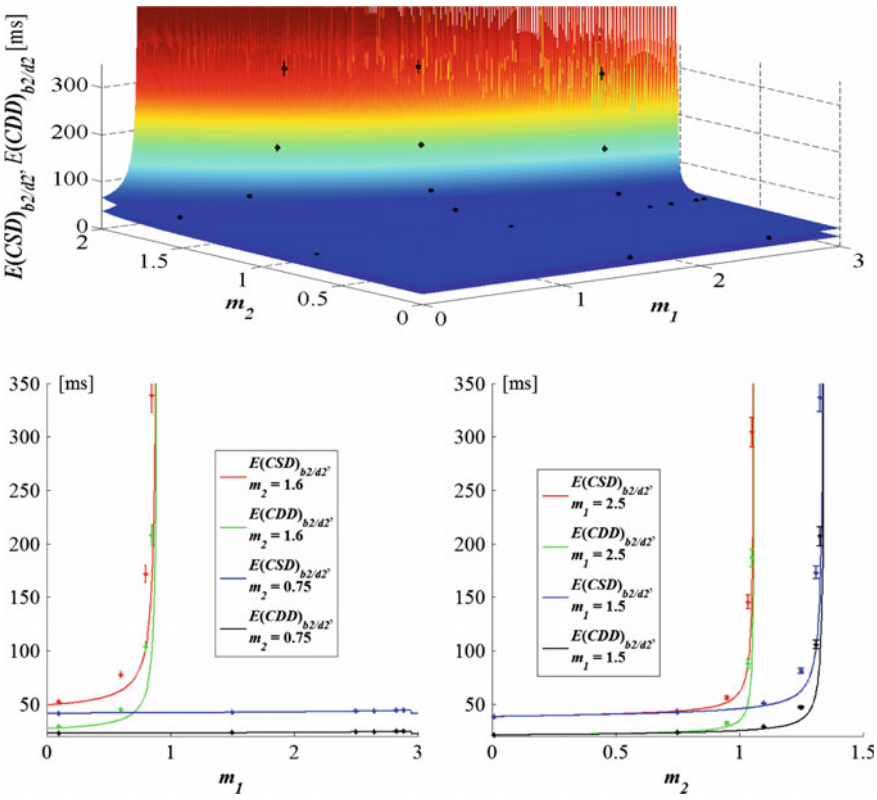

Fig. 12 Verification of the analytical results for data set 2 from [6] (Table 2)—intra-domain calls (scenario b2/d2)

References

1. General overview of NGN, ITU-T Recommendation Y.2001, Dec 2004
2. IP Multimedia Subsystem (IMS); Stage 2 (Release 12), 3GPP TS 23.228 v12.4.0, Mar 2014
3. Call processing performance for voice service in hybrid IP networks, ITU-T Recommendation Y.1530, Nov 2007
4. SIP-based call processing performance, ITU-T Recommendation Y.1531, Nov 2007
5. Kaczmarek, S., Sac, M.: Traffic model of a multidomain IMS/NGN. Telecommun. Rev. Telecommun. News. **8–9**, 1030–1038 (2014)
6. Kaczmarek, S., Sac, M.: Call processing performance in a multidomain IMS/NGN with asymmetric traffic. In: Grzech, A., et al. (eds.) Information Systems Architecture and Technology: Selected Aspects of Communication and Computational Systems, pp. 11–28. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław (2014)
7. Kaczmarek, S., Sac, M.: Traffic model for evaluation of call processing performance parameters in IMS-based NGN. In: Grzech, A., et al. (eds.) Information Systems Architecture and Technology: Networks Design and Analysis, pp. 85–100. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław (2012)

8. IMS for next generation networks, ITU-T Recommendation Y.2021, Sept 2006
9. Pirhadi, M., Safavi Hemami, S.M., Khademzadeh, A.: Resource and admission control architecture and QoS signaling scenarios in next generation networks. World Appl. Sci. J. **7**, 87–97 (2009). (Special Issue of Computer & IT)
10. OMNeT++ Discrete Event Simulator—Documentation.: https://omnetpp.org/documentation
11. Rosenberg, J., et al.: SIP: Session Initiation Protocol, IETF RFC 3261, June 2002
12. Calhoun, P., et al.: Diameter Base Protocol, IETF RFC 3588, Sept 2003
13. Kaczmarek, S., Kaszuba, M., Sac, M.: Simulation model of IMS/NGN call processing performance, vol. 20, pp. 25–36. Gdańsk University of Technology Faculty of ETI Annals (2012)
14. Abhayawardhana, V.S., Babbage, R.: A traffic model for the IP multimedia subsystem (IMS). In: IEEE 65th Vehicular Technology Conference (VTC2007), Dublin, Ireland (2007)
15. Kaczmarek, S., Sac, M.: Analysis of IMS/NGN call processing performance using phase-type distributions. In: Grzech, A., et al. (eds.) Information Systems Architecture and Technology: Network Architecture and Applications, pp. 23–39. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław (2013)
16. Kaczmarek, S., Sac, M.: Analysis of IMS/NGN call processing performance using phase-type distributions based on experimental histograms. Int. J. Commun. Syst. (2016). (Submitted for publication)

A Novel Approach to Automating Operating System Configuration Management

Błażej Święcicki

Abstract In the last few years, operating system configuration management tools have seen a dramatic rise in popularity. The reason is that system administrators need to operate on a higher number of servers than before, and because of that, they need tools to automate some of their work. This work summarizes the advantages and disadvantages of existing tools, and proposes a novel approach to operating system configuration management. The author defines a new programming language for describing an operating system configuration, and then describes a method for producing programs from a source file written in this language. These programs make appropriate changes to the system's state, applying the desired configuration. This approach allows for features not present in other tools, while also potentially consuming less resources, and having less runtime dependencies.

Keywords Operating system configuration management · Programming languages

1 Introduction

Operating system configuration management tools assist a system administrator in automating significant parts of his work. Many system administrators configure their systems either manually, or using a custom shell script [1]. However, because of the rising popularity of virtualization, and lately, containerization [2], automating operating system configuration is more desirable, and in some cases, necessary. Unfortunately, due to a large number of possible errors and general complexity of these tasks, shell scripts become quickly unmaintainable. Other authors use the term *Software Configuration Management* [3] *System Configuration* [4], or *Configuration Management System* [5], however these might be ambiguous. In order to avoid that, the author uses *Operating System Configuration Management*

B. Święcicki (✉)
Politechnika Wrocławska, Wrocław, Poland
e-mail: blazej.swiecicki@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_10

as a common name for the category of software described in this work, but all of these names might be used interchangeably when there is no ambiguity.

In this work, the author briefly examines existing solutions to this problem and identifies some of their advantages and disadvantages. A new tool is then proposed to address the previously shown problems of existing software.

2 Related Work

The author has chosen four widely used operating system configuration management tools for comparison: CFEngine, Puppet, Chef and Salt. The first three have been previously compared by other researchers in [3, 4] using different criteria. Salt was also compared to the previous three in [5].

Of all the tools compared, only CFEngine is has been described in detail in academic work. It's based on the promise theory [6] developed by Burgess in 2005 [7].

CFEngine is the most scalable tool of the ones compared here [4], and requires the lowest number of dependencies [3]. However, it's restricted to a limited set of basic operations. It's not possible to extend that set, so any operations not supported by CFEngine must be carried out by external programs.

Puppet [8] is a newer tool that partially addresses this concern, at the cost of being more resource-intensive [5] and requiring more dependencies [3]. In contrast with CFEngine, which is written in C, Puppet is written in Ruby, which contributes to its resource usage.

Chef requires the most dependencies [3] and has a steep learning curve [5], making it hard to use [4]. That said, Chef is the most flexible tool of the ones compared, because its configuration is written in Ruby—a general purpose programming language. That feature by itself is what makes it both flexible and hard to use, because it requires using an imperative programming language. In contrast to this, all the other configuration management tools described in this work use a declarative syntax for specifying configuration.

Salt is a simpler tool [5] that is set apart from the other ones by the fact that its configuration can be written in any programming language, as long as the program outputs a valid data structure. In simpler configurations, it can also be specified in JSON or YAML.

All of these configuration management tools operate in a similar manner: there is a master server that distributes configuration to its agents. Agents are programs that receive the configuration from a master server, and apply it on the target machine.

The author of this work proposes a different approach. The configuration management tool described in this work will not enforce a configuration distribution strategy at all. Instead, it will generate a standalone computer program that applies the desired configuration. This program then could be transported to the target machine, and executed there. By using this approach, the author has managed to avoid requiring any dependencies for the generated programs, making them quite small and

self-contained, which in turn makes them more suitable for use in embedded systems and other scenarios where installing additional software is undesirable.

Another issue in system configuration management is extensibility. As seen above, some of the existing tools have a predefined set of capabilities that cannot be extended. This is undesirable, and so, the proposed system will be as extensible and flexible as possible. This flexibility, unlike in Chef, must not come at the cost of drastically increased complexity. This choice precludes usage of a general purpose programming language for configuration.

In summary, there are 5 attributes considered: number of dependencies on the configured server, configuration language paradigm, language of implementation for the agent program, extensibility (whether the user can create new types of resources), and data processing ability (whether one can configure the server using data from the target server). For the compared tools, the results are as follow.

3 Design and Architecture of the Proposed Tool

This work describes a new operating system configuration management tool created by the author, called Overlord. As has been mentioned above, it takes a different approach to applying the configuration than other tools—instead of using an agent program that receives instructions from a master server, Overlord creates a standalone program that independently applies the previously specified configuration.

For this purpose, Overlord defines its own programming language.

3.1 The Programming Language Used by Overlord

The language described here aims to be both simple and flexible—its design is influenced mainly by Puppet, UCL [9] and Lisp. A program in this language is a sequence of resources. A resource:

- Has a type.
- May have a name.
- Has 0 or more attributes. An attribute is a pair (name, value).

In Puppet, resources represent operating system entities, like users, services, etc. [8]. Overlord's notion of resources is more generic. A resource may represent an operating system entity, but may also be used to provide context for other resources or even used to control the flow of the program. A simple resource is written as follows:

```
user john {}
```

The above example is a complete, valid Overlord program that ensures a presence of an operating system account named "john". In this case, "user" is the

resource's type, and "john" is its name. The braces at the end of the line may contain attributes:

```
user john {
uid = 1001
gid = 100
}
```

This example extends the previous one by adding two attributes to the resource, named "uid" and "gid". The modified program would still ensure that there is a user named "john", but there wasn't, it would create it with the specified user id (uid), and group id (gid).

Resources can also provide context for other resources. This kind of usage has two main variants.

```
sudo john {
body = {
shmkdir ~/.ssh {}
     }
}
```

The first example, above, shows two resources, "sudo" and "sh", the first containing the latter. Resources of type "sudo" execute their inner resource with the credentials and environment of another user. This is useful for resources like "sh", which use their environment to infer some information. In the example above, the "sh" resource creates a directory ".ssh" in the home directory of the user "john". Without the "sudo" resource, the directory would be created in the home directory of whoever would execute the program. This example also shows that an attribute's value may be a set of resources. Puppet and other tools do not allow this.

The other variant is a resource that provides a namespace for its inner resources:

```
nginx.site default {
     location / {
         proxy = "http://127.0.0.1:3000";
     }
}
```

In the above example, a resource of type "location" is used, despite that there is no such resource type defined. However, the outer resource, "nginx.site", prepends the string "nginx.site." to all names of types contained inside it, effectively creating a namespace. Because, of that, the inner resource type is transformed to "nginx.site.location" that exists and configures a location for a "nginx.site" resource. This example also shows that resources may directly contain other resources, without any attributes—in such a case, these resources are appended to the, possibly empty, list of resources in the "body" attribute of the outer resource.

Yet another possible usage of resources is providing control flow. One example is a for loop:

```
for {
var = i
in = range(5)
shconcat(touch, $i) {}
}
```

A resource with type "for" executes the contents of its "body" attribute, each time assigning a different value to a variable with name specified by "var". The values are specified in the "in" attribute.

The above example introduces two important language constructs: function calls and variables. Function calls—shown as the value of the "in" attribute and the name of the "sh" resource—provide an intuitive way to process data and gather information at run time. Variables are simply a syntax sugar: `$x` is changed into a function call `var(x)`.

Functions are also supported in Puppet, but with a major difference. In Puppet, functions are evaluated on the server—Overlord can evaluate them both during compilation and at run time.

Functions can take 0 or more arguments, separated by a comma. These arguments can be any value, or another function call, but not a set of resources.

An important feature of the language is that resources have complete control over their contents. As seen above, a resource can choose how, and how many times to execute its inner resources, it can change their type, or perform other changes, as necessary. Functions take arguments in a similar manner—they have complete control over them. One example of how this can be useful is a function to produce JSON objects:

```
json(
    key(a, 2),
    key(b,
        key(c, 3),
        key(d, 5)
    )
)
```

This function checks its arguments instead of trying to evaluate the "key" function calls, transforms them to entries in a data structure. The result of this particular call is a JSON document:

```
{"a": 2,
 "b": {
   "c": 3,
   "d": 5
 }
}
```

This language doesn't have built-in constructs for control flow, assignment, variables, or definition of resources and functions. This is by design—the language is small, but flexible enough to delegate this functionality to libraries. For example,

the author has defined a function “lambda” that returns a function to be used elsewhere. An example usage:

```
filter(lambda(link, starts_with($link), "proxy:")),
$links)
```

This example filters a list, selecting only elements whose values start with “proxy:”. It demonstrates that the functional style of programming may be used in this language.

3.2 Architecture

In order to produce a standalone program from a file in the source language, Overlord compiles the source to another language, and then compiles the resulting file to a binary. There can be various target languages for compilation, including Bash, C, and for example, Puppet’s and CFEngine’s languages. The author has chosen the Rust programming language [10] for the initial implementation. There are two main reasons for this choice. First, it’s a compiled language, which is a requirement for generating standalone programs. Second, the safety guarantees it provides ensure that the compiled program will not crash unexpectedly. Nevertheless, Overlord does not depend on any particular target language, and support for other languages can be added later.

As seen in Fig. 1, the process of compilation from Overlord’s language to the target language is divided into three stages—parsing, compilation, and building. First, the code is parsed into a data structure that is later passed between Overlord and its modules. The second stage transforms this data structure into chunks of target language code, along with metadata. Lastly, the chunks of target language code are joined into one file, and the metadata is used to generate necessary headers and definitions. While the first and third stages are rather simple, the second one requires a more detailed description.

After parsing the source code, Overlord has to transform it to the target language. In order to do so, it must know how to transform each resource and function into code in this language. In order to be extensible, all resources and functions are implemented in modules separate from the main program. A module can be written in any programming language—it is executed as a separate program, and communicates with Overlord through the standard input and output streams.

For each resource and function, overlord finds the appropriate module, launches it, and then requests that the module transforms a resource into code in the target language. All communication between the main process and modules is done using JSON documents, one document per message. There are three possible kinds of messages:

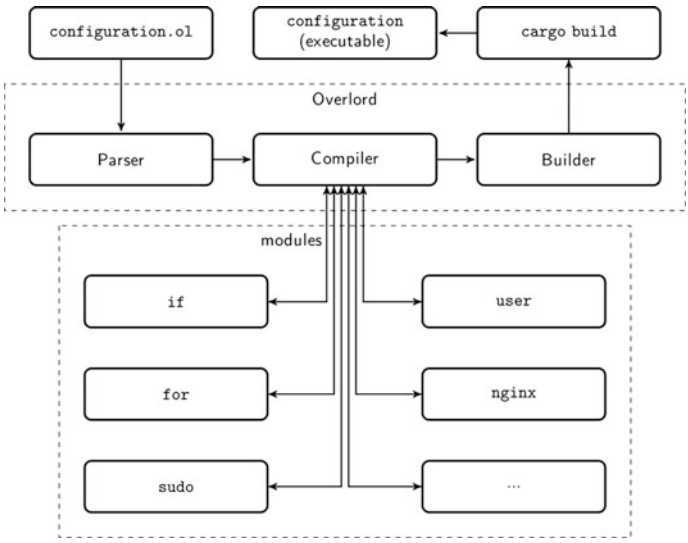

Fig. 1 An overview of overlord's architecture

- request,
- response,
- metadata.

The last one is used to inform the main process about the resources and functions a given module supports. Some modules may want to support several resources or functions, and this information prevents Overlord from having to execute multiple copies of the same module unnecessarily.

Once launched, the modules' processes are not terminated until the end of compilation. This is done in order to minimize the overhead from executing a new process, which was found to be significant.

When a module encounters a resource or function that it doesn't know how to compile into code, it can request the compilation from the main process, which will forward it to the appropriate module and handle communication. The main process may, however, return another request instead of a response. This will happen, for example, if a resource contains another resource of the same type. Because one process is reused for all requests for the same type of resource, it cannot be guaranteed that the requesting module will not be interrupted by another request.

A response is usually a code fragment with associated data, such as function definitions, or compile-time dependencies. The main process of Overlord extracts this associated data in order to pass it to the second stage of compilation. If a module has requested a given fragment of code, it usually incorporates it into its

own code, and returns it as a response for the main process. If the main process was the one requesting, as is the case with top-level resources, it appends the fragment to the list of code fragments passed to the second stage. The reason for splitting this process in two stages is that the second one depends on the target language. The process of preparing the final source code in from code fragments might be different for Rust, and for example, Bash.

The source in the target language is then passed to a compiler for the target language, if necessary. In case of Rust, the resulting program depends only on the C library, available on practically every operating system in use.

4 Results

In order to check whether Overlord realizes its goals, the author has described three scenarios in which it might be useful, and then attempted to use Overlord in each of them. This section focuses on these scenarios.

4.1 The First Scenario—Installing Overlord Itself

Since Overlord generates standalone programs, and its capabilities must include installation of software, it should be also possible to generate a working installer for the tool.

While programs generated by Overlord don't have external dependencies, the tool itself does. The current implementation is written in Ruby, so it requires a Ruby interpreter. In order to compile programs, it also needs a Rust compiler. Overlord can be installed as a Ruby gem,[1] automatically installing any other necessary dependencies.

Such a program is easily created in Overlord. It is also easy to perform the same tasks using CFEngine, Puppet, or Salt, but these tools can't generate a standalone installer—only perform the tasks on a machine where their agent program is installed.

The program source code itself is not included due to space constraints, however it consists of only 16 lines of code comprising 529 bytes.

The author has verified that the generated installer indeed works as intended, installing a working copy of Overlord on a target machine. This copy was also verified to be able to compile the installer program.

[1] Ruby packages are called gems.

4.2 The Second Scenario—A Shared Hosting Environment

The purpose of many servers is to serve web pages or web applications to many users. Administrators of such systems must provide a way for the users to configure the web server to serve their applications. One method used in practice is to have a script scanning for specific symbolic links in the users' home directories, and then generate the web server configuration based on these links. An example of such a symbolic link would be: `www.example.org -> proxy:127.0.0.1:3000`. These links have invalid destination paths, which instead of being a path to a file, contain configuration data for the script mentioned above. This particular example would configure the web server to act as a reverse proxy for "www.example.org", serving data from a different web server at "127.0.0.1:3000" for the domain.

Neither CFEngine, Puppet, not Salt are able to handle this task, as they are not capable of processing data from the target system. One of the design goals of Overlord, however, was to make if flexible enough for such tasks.

The author has written a program that demonstrates that it is indeed possible to realize this task with Overlord. The source code for the program is 25 lines long, totaling 680 bytes.

This result is important especially because it shows that Overlord is able to do more than the other configuration management tools, without resorting to using a general-purpose programming language for configuration.

4.3 The Third Scenario—Installing a Web Application

A common task for a configuration management tool is to install and configure a web application. The author has decided to do that as well, using Overlord. The application chosen was Redmine, a popular project management tool. Its installation process is quite complex.

It should be run from a separate user account, so that account has to be created and most other actions have to be carried out as the "redmine" user. Redmine is a Ruby application, so it needs a Ruby interpreter installed, along with the libraries it depends on. Ruby is installed as the "redmine" user, using the Ruby Version Manager (RVM).

Most of the commands that have to be executed must be launched in the context of RVM—otherwise they won't find the Ruby interpreter and will fail. Because of that, the author has defined an "rvm" resource type that ensures the Ruby Version Manager, and a correct version of ruby are installed, then executes commands in the proper context.

Aside from that, a database configuration file has to be created, and the database itself must be created and initialized. Additionally, a system service has to be created and started.

The source code for this scenario is 67 lines long—mostly because of the amount of tasks to perform. It has 1790 bytes.

4.4 Testing Methodology

The programs written and compiled for the above scenarios were tested on a virtual machine, to ensure that the configuration was applied fully automatically. Separate virtual machines were created for each test. The operating system in each of the machines was a fresh installation of Debian Jessie GNU/Linux, without any post-installation user intervention.

5 Conclusions

After examining existing operating system configuration management tools, the author has concluded that all of them were either not extensible enough, hard to use, or required too many dependencies on a target machine. Therefore, the author has proposed a new tool that addresses these concerns and represents a new approach to the problem.

In terms of Table 1, Overlord's has the following properties:

- Dependencies: 0
- Configuration language: declarative with functional elements
- Language of implementation (for agent): Rust
- Extensibility: yes
- Data processing ability: yes

It generates standalone binary programs that may then be uploaded to the target machine and executed there. The problem with dependencies is solved this way, since the programs do not have any.

Ease of use is hard to measure, but Overlord strives to achieve that by using a domain specific language for its configuration. Because the user does not have to

Table 1 Properties of the compared configuration management tools

	CFEngine	Puppet	Chef	Salt
Dependencies	3	11	67	35
Language paradigm	Declarative	Declarative	Imperative	Declarative
Implementation language	C	Ruby	Ruby	Python
Extensibility	No	Yes	Yes	Yes
Data processing ability	Limited[a]	Limited[a]	Yes	Limited[a]

The values for number of dependencies for CFEngine, Puppet, and Chef are taken from [3]
[a]Can only use predefined, primitive values, like an IP address of a network interface

learn a full, general purpose programming language to use Overlord, it is easier to use than, for example, Chef.

Extensibility is central to the idea of this tool. It can be extended in various ways. Firstly, new target languages can be added. If for example, a Bash script would be desired instead of a binary program, this can be done by adding support for Bash as the target language for Overlord.

Secondly, new resources and functions can be defined. An important feature is that they can be implemented in any programming language. This allows system administrators, who are usually not full-time programmers, write new resources and functions in a language they are proficient in, without having to learn a new one just to extend Overlord.

And finally, Overlord gives complete control of the evaluation of resources' and functions' contents, allowing them to implement functionality usually reserved to the language itself, like loops, conditionals, and namespaces.

These features make Overlord a unique tool with a lot of potential for future development.

6 Future Work

There are three main areas that the author has identified for possible further research.

A Type System for Overlord Currently, all variables and data are weakly typed—their interpretation is up to the resource or function that uses it. Weak typing is, however, often regarded as inferior to its alternative. A strongly typed variant of Overlord's language might create possibilities for better error reporting, or automatic iteration over arrays. However, it requires further investigation whether such a variant would be practical.

Dependencies Between Resources Unlike other configuration management tools, Overlord provides no way of specifying dependencies between resources. Because resources in Overlord may contain other resources, however, implementation of such a feature might not be trivial.

Generating a Program in Multiple Languages In some cases, a task can't be implemented in the target language, or there is simply no module for that language available. In these situations, Overlord could generate a separate program in another language, and bundle it with the main program. This could be especially useful if the target language was constrained, as is the case with CFEngine. Overlord could generate a CFEngine configuration file where possible, and create separate programs for the parts which CFEngine can't handle itself.

References

1. Burgess, M.: Cfengine: a site configuration engine. In: USENIX Computing Systems, vol. 8, No. 3 (1995)
2. Turnbull, J.: The Docker Book: Containerization is the New Virtualization (2014)
3. Önnberg, F.: Software Configuration Management: A Comparison of Chef, CFEngine and Puppet. University of Skövde, School of Humanities and Informatics (2014)
4. Delaet, T., Joosen, W., Vanbrabant, B.: A Survey of System Configuration Tools. LISA (2010)
5. Torberntsson, K., Rydin, Y.: A Study of Configuration Management Systems: Solutions for Deployment and Configuration of Software in a Cloud Environment. TVE, 14 013 (2014)
6. Burgess, M.: Some Notes About Promise Theory. http://markburgess.org/PromiseMethod.pdf (2015)
7. Burgess, M.: An approach to understanding policy based on autonomy and voluntary cooperation. Lect. Notes Comput. Sci. **3775**, 97–108 (2005)
8. Kanies, L.: Puppet. LISA (2006)
9. Stakhov, V.: https://github.com/vstakhov/libucl
10. http://www.rust-lang.org/

A Study on the Effectiveness of Threshold and Traffic Overflow Mechanisms in Multi-service Switching Networks with Real-time and Non-real-time Traffic

Mariusz Głąbowski and Michał Dominik Stasiak

Abstract This article presents the results of a simulation study that examines the performance and the effectiveness of threshold mechanisms and traffic overflow mechanisms in multiservice switching networks. A threshold mechanism is used in the study to control those services that do not require data transmission in real time, i.e. the so-called elastic services. Traffic streams that are generated by these services (non-real-time traffic) can have variable transmission time dependent on available resources in the system. The traffic overflow mechanism is used, in turn, to guarantee the delivery of real-time traffic streams for which the amount of demanded resources cannot be reduced. The study confirms the effectiveness of the application of the proposed mechanisms in decreasing the call blocking probability in multi-service switching networks.

Keywords Multiservice switching networks · Threshold mechanism · Overflow links · Elastic traffic

1 Introduction

Efficient operation of modern-day telecommunications and computer networks is largely dependent on the performance and effectiveness of traffic management in nodes of telecommunications and computer networks. These devices are based on multi-service switching networks, i.e. networks that service a mixture of traffic streams of different classes with differentiated parameters of sources that generate these streams, and with differentiated service demands related to these streams in the network [1]. To ensure a differentiated service quality of traffic it is necessary to

M. Głąbowski (✉) · M.D. Stasiak
Faculty of Electronics and Telecommunications, Poznan University of Technology,
ul. Polanka 3, 60-965 Poznan, Poland
e-mail: mariusz.glabowski@put.poznan.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_11

introduce appropriate traffic management mechanisms in switching networks [2–4]. One of the most widely used traffic management mechanisms is the so-called threshold mechanism [3–5]. This mechanism is capable of limiting the amount of resources allocated to given traffic streams admitted for service—after a given defined load threshold for as system is exceeded—and simultaneously lengthen their service time (e.g. a file transfer for which all involved data transfer is essential), that is to streams that are generated by the so-called elastic services.

Threshold mechanisms require a single threshold to be introduced to the system in STS (Single Threshold System) systems [3–5], or a number of thresholds in MTS (Multi Threshold System) systems [3, 4, 6, 7]. Feasibility of the application of threshold mechanisms in multi-service switching networks is demonstrated and discussed in [8–10]. The results obtained in the studies clearly indicate that a large decrease in the internal blocking probabilities of elastic traffic streams that are to be optimized by threshold mechanisms is possible.

The threshold mechanism can be employed for elastic services, i.e. non-real-time services. However, a high percentage of traffic streams offered to switching networks is composed of streams of real-time services for which the service time and the volume of allocated resources cannot be changed. To limit the internal blocking probability for real-time traffic streams—in multi-service switching networks—an overflow mechanism for traffic overflow between neighboring switches of a selected stage of a switching network can be used. This mechanism requires only a slight intervention into the structure of the network, which is based on an introduction of an additional link between switches of the same stage. Overflow links were introduced for the first time to Pentaconta switching nodes manufactured between 1960 and 1980 [11]. The possibility of an introduction of overflow links was also considered in digital switching networks [12–14]. The studies on the application of overflow links in present-day network devices [15–18] that make use of networks servicing multi-service traffic have proved that the application of overflow links in these networks is followed by a significant increase in their traffic effectiveness as a result of a decrease in the internal blocking probability.

The application of the traffic overflow mechanism makes it possible to decrease the internal blocking probability for traffic classes that undergo this mechanism, without any necessity to decrease resources allocated to them. This mechanism can be then used for traffic classes that require real-time data transfer, but at the same time a decrease in the intensity of data streams generated by them is not possible.

Until now, threshold mechanisms and the overflow mechanism have been addressed in the literature of the subject separately. In the present article switching networks in which to decrease the blocking probability of real-time traffic streams the traffic overflow mechanism will be used, whereas to decrease the blocking probability for non-real time traffic (elastic services)—the threshold mechanism will be used, are considered for the first time. The effectiveness of these mechanisms will be investigated with the help of simulation methods for different structures of offered traffic and different structures of switching networks.

The article is structured as follows. Section 2 presents the structure and the operation of the switching network with overflow links and threshold mechanisms.

Section 3 presents the results of the simulation study for different scenarios for the application of threshold mechanisms and traffic management mechanisms that direct traffic to overflow links. Section 4 sums up the results of the study.

2 Multi-service Switching Networks Witch Overflow Links and Threshold Mechanisms

Let us consider now the three-stage Clos switching network servicing multi-rate traffic streams presented in Fig. 1. Each of the switching modules in each stage includes n symmetrical switches with $n \times n$ links.

All the input, output and inter-stage links in the network have the same capacity equal to f BBUs (Basic Bandwidth Units [4, 19]). The output links have been grouped into the so-called output directions in such a way that the first output link of each of the switch of the last stage forms the first direction, and so on. The assumption adopted in this study is that the overflow links will be used in the first stage of the network and that will connect the additional output of a given switch with an additional input of a neighboring switch in the first stage (Fig. 1). Another assumption is that the considered network operates in the point-to-group selection mode, i.e. the control algorithm always tries to set up a connection of a given call class between a given input link and a free output link of the demanded direction.

The switching network is offered a mixture of M traffic classes. Each class is characterized by its own call intensity λ_i, service intensity μ_i, and demands of t_i BBU to set up a connection. In the considered switching network the external and the internal blocking can occur. The phenomenon of the external blocking occurs when all output links in a given direction are busy, whereas the phenomenon of the internal blocking appears when a connection between a free output and a free input cannot be set up due to the occupancy of inter-stage links in the network.

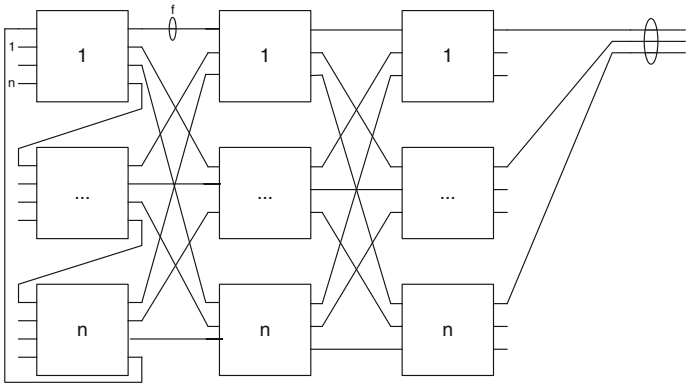

Fig. 1 Three-stage Clos network with overflow links

2.1 Operation of the Overflow Mechanism in the Switching Network

In the case of the point-to-group selection, upon an appearance of a new call of a given traffic, the control algorithm chooses a last-stage switch that has a free link in the demanded direction (i.e. one with the number of free resources that would allow a given connection to be carried). Then, the algorithm attempts to set up a connection to the chosen switch of the last section. If this is not possible, the algorithm —for calls of those traffic classes for which the application of traffic overflow is feasible—will attempt to set up a connection via an overflow link, i.e. the call will be forwarded to a neighboring switch of the first stage and an attempt to set up a connection between the switch of the first stage and the selected switch of the last stage will be made. If this connection fails to be established, the algorithm will choose another last-stage switch that has a free output link in the demanded direction and will repeat the above sequence of attempts to set up a connection. In the case when no connection to free switches of the last stage can be set up, this call will be lost due to the internal blocking.

The traffic overflow mechanism applicable in the considered switching network is an effective mechanism for decreasing the internal blocking probability for non-adaptive real-time traffic streams. This solution does not add extra load to the control device and its cost can be reduced to just an application of switches with one additional input and output in the first stage of the network.

Simulation studies [15] confirm that the application of overflow links in the first stage of the network appears to be the most effective. Due to technical reasons, the capacity of overflow links is equal to the capacity of inter-stage links. When this is the case, the application of the overflow mechanism leads to a decrease in the internal blocking probability by from 50 to 80 %. Analyses of switching networks [15, 16] show that an increase in the capacity of overflow links above the capacity of the inter-stage link has only a slight impact on the value of the internal blocking probability and, in the case of the point-to-group selection, this influence is neglectable. The application of overflow links is possible both in the case of electronic networks with a large number of inputs and outputs, and in smaller but much faster networks based on optical technologies. The introduction of overflow links does not add load to the control algorithm and, with just minor costs of a change to the structure of a network involved, makes it possible to significantly decrease the internal blocking probability in the network.

2.2 The Operation of the Threshold Mechanism in the Switching Network

Traffic engineering distinguishes between STS systems (Single Threshold Systems) and MTS systems (Multi-Threshold System) [3–7]. Thresholds introduced to the

system, defined by the total number of busy BBUs, divide the system into two (in the case of STS) or more (in the case of MTS) threshold areas. The number of BBUs, allocated by the control function and necessary to service a call of a given class, depends on the load of the system, i.e. on the threshold area in which the system is in. As the load in the system increases, the number of BBUs allocated to calls of particular classes decreases, while the service time is simultaneously increased, proportionally, to ensure that all data can be transmitted. Figure 2 shows the operation of a dual threshold system. If the load of the system (measured by the number of occupied BBUs) is lower than the demanded threshold $Q_{i,1}$, then the control function will allocate to an incoming call of class i the maximum bit rate, expressed in BBUs and equal to $t_{i,0}$. An increase in the load of the system exceeding the threshold $Q_{i,1}$, results in the allocation of a lower number of BBUs, equal to $t_{i,1}$.

Each traffic class that undergoes the threshold mechanism will be described by the following parameters:

λ_i intensity of calls of class i,

$\mu_{i,k}$ service intensity of calls of class i in the threshold area between thresholds k and $k + 1$, where $0 \leq k \leq q_i$

q_i number of thresholds introduced to system for traffic of class i,

$Q_{i,k}$ value of threshold k for class i traffic, expressed in BBUs, where $Q_{i,0} = 0$

$t_{i,k}$ number of BBUs, allocated to calls of class i necessary to set up a connection within a threshold area between thresholds k and $k + 1$, where $0 \leq k \leq q_i$

Simulation and analytical studies [8–10] confirm the feasibility of applying threshold mechanism in output links in multi-service switching networks. The introduction of threshold mechanisms for a number of selected call classes of elastic traffic makes it possible to significantly reduce the blocking probability of these traffic classes, leading at the same time to an increase in the traffic capacity of the network.

In the remainder of the article the results of simulation studies of a number of selected multi-service switching networks in which overflow links for real-time traffic classes have been introduced will be discussed, as well as threshold mechanisms for elastic traffic classes not related to real-time applications.

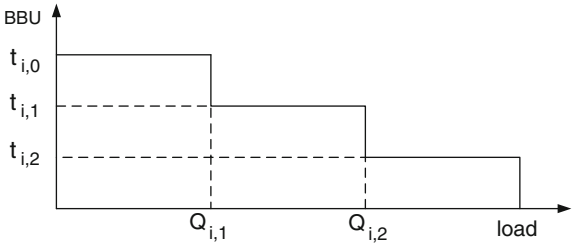

Fig. 2 Double threshold mechanism for class i calls

3 Simulation of Multi-service Switching Networks

To determine the traffic effectiveness of multi-service switching networks with overflow links and threshold mechanisms implemented in output links, a dedicated simulator was devised. The simulator allowed traffic characteristics for all traffic classes serviced in the network to be determined with a demanded value of the confidence interval. To construct the simulator, the discrete event method was used [15, 20]. The simulator made it possible to investigate Clos switching networks with the point-to-group selection and with overflow mechanisms introduced to the first stage of the network and threshold mechanisms introduced in the output links.

3.1 Assumptions Adopted in the Simulation Experiments

A three-stage Clos switching network was chosen for the simulation study (Fig. 1). The network was composed of switches with $n \times n$ links, each link with the capacity of f BBU. The assumption was that $n = 4$, and $f = 30$ BBU. One output direction had the capacity of $V = nf = 120$ BBUs. The overflow links were introduced to the first stage. Each switch in this stage had one additional input and output (the first-stage switches had therefore the size: 5×5). The overflow links connected then the additional, the fifth, output of a given switch with an additional, the fifth, input of the neighboring first-stage switch (Fig. 1).

The assumption is that the multi-service switching network is offered a mixture of different Erlang traffic classes. This means that a traffic stream of each of the traffic classes is a Poisson stream, whereas the service time of each call class is described by exponential distribution. The simulator employed a random algorithm for setting up connections—from among all possible connection paths between a free input and a free output, a single connection path is randomly selected to execute a given connection. The simulator was written in the C++ language and allowed the blocking probability for each call of class i ($1 \le i \le M$ to be recorded, where M is the number of traffic classes offered in the switching network):

$$E_i = L_{i,t}/L_{i,o} \tag{1}$$

where $L_{i,t}$ is the number of lost calls of class i, whereas $L_{i,o}$ is the number of offered calls of class i within the considered period.

According to the observations given in Sect. 2, traffic classes that do not undergo the threshold mechanism can be serviced by overflow links. The network is offered three traffic classes in the following proportion: $A_{1,0}t_{1,0} : A_{2,0}t_{2,0} : A_{3,0}t_{3,0} = 1{:}1{:}1$, where $t_{i,0}$ is the number of BBU allocated to calls of class i to set up a connection in the first threshold area, i.e. between thresholds 0 and 1, while the parameter $A_{i,0}$ is offered traffic of class i:

$$A_{i,0} = \frac{\lambda_i}{\mu_{i,0}}. \tag{2}$$

Let us assume that traffic of class i is an elastic traffic and load of the system exceed the threshold $Q_{i,k}$. Under these circumstances, the decrease in required number of BBUs from $t_{i,0}$ BBUs to $t_{i,k}$ BBUs causes the increase in service time of class i calls from $\tau_{i,0}$ to $\tau_{i,k}$:

$$\tau_{i,k} = \tau_{i,0} \frac{t_{i,0}}{t_{i,k}}, \tag{3}$$

where $\tau_{i,0}$ is the service time of a given class i call obtained from random number generator of exponential distribution with parameter $\mu_{i,0}$.

The simulation experiments were conducted for such a number of series (each series comprised of 100,000 calls of each traffic class) that would ensure a 99 % confidence interval determined on the basis of the t-Student distribution. In the conducted simulation experiments this interval is at least by one order of magnitude lower than the average value for the simulation results. The confidence interval is not always marked in the graphs because in some cases it is lower than the graphic symbol denoting the result of the simulation. The results of the simulation experiments are presented in Figs. 3, 4, 5 in relation to the maximum intensity of traffic offered to one BBU of an output link (i.e. of traffic defined with the assumption that elastic traffic is determined by the parameters for the first threshold area):

$$a = \sum_{i=1}^{M} \left(\frac{A_{i,0} t_{i,0}}{n^2 f} \right). \tag{4}$$

3.2 *Results of the Simulation Study*

The assumption in the first simulation experiment was that the first and the second traffic class were classes of elastic traffic and underwent the threshold mechanism. The first class of elastic traffic (demanding respectively 10 BBUs to set up a connection in the first threshold area) make use of the dual-threshold mechanism. The second class (demanding respectively 6 BBUs to set up a connection in the first threshold area) makes use of the single-threshold mechanism.

Another assumption was that the third traffic class did not undergo the threshold mechanism (real-time traffic) and could be serviced via overflow links. All the traffic classes offered in the switching network in this experiment are described by the following parameters, expressed in BBUs:

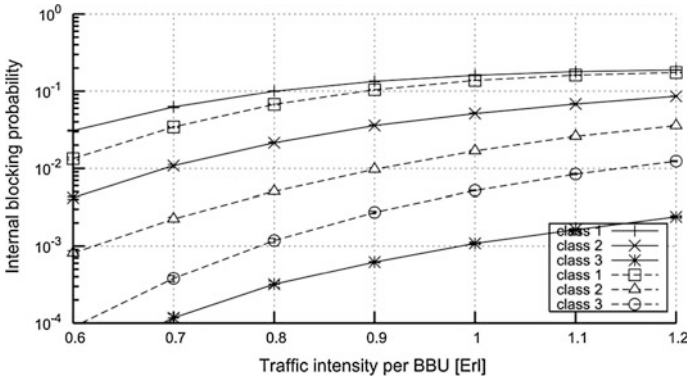

Fig. 3 The internal blocking probability in the switching network, *solid lines*—switching network without threshold mechanism, *dotted lines*—switching network with threshold mechanism

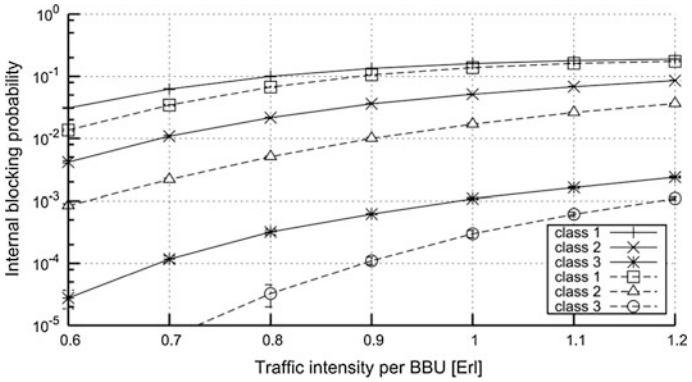

Fig. 4 The internal blocking probability in the switching network, *solid lines*—switching network without threshold and overflow mechanisms, *dotted lines*—switching network with threshold mechanism for classes 1 and 2, and overflow mechanism for class 3

$q_1 = 2$, $Q_{1,1} = 80$, $Q_{1,2} = 100$, $t_{1,0} = 10$, $t_{1,1} = 7$, $t_{1,2} = 5$,
$q_2 = 1$, $Q_{2,1} = 80$, $t_{2,0} = 6$, $t_{2,1} = 3$,
$q_3 = 0$, $t_{3,0} = 2$.

The results of the simulation of the internal blocking probability are presented in Fig. 3. The introduction of the threshold mechanism for elastic traffic classes (classes 1 and 2) results in a decrease in the internal blocking probability for these classes of calls. However, the blocking probability for the third class increases. This effect can be explained as follows. An application of the threshold mechanism is followed by an increase in load generated by elastic streams. As a result, free resources for the third class calls are reduced and, in consequence, blocking probability of this class increases.

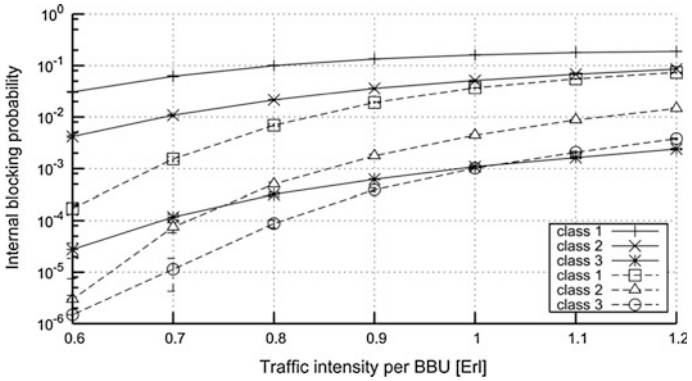

Fig. 5 The internal blocking probability in the switching network, the *solid line*—switching network without threshold and overflow mechanisms, the *dotted line*—switching network with threshold mechanism for classes 1 and 2, and overflow mechanism for all classes

The assumption in the second simulation experiment was that the first and the second traffic class were classes of elastic traffic and underwent the threshold mechanism. Another assumption was that the third traffic class (real-time traffic) could be serviced via overflow links. All the traffic classes offered in the switching network are described by the parameters adopted in the previous experiment. The results of the simulation of the internal blocking probability are presented in Fig. 4. The results indicate high effectiveness of the application of the overflow link system and the threshold mechanism in the switching network under consideration. The introduction of the overflow mechanism for real-time calls (class 3) leads to a significant decrease in the internal blocking probability in this call class (one order of magnitude), and virtually does not influence the internal blocking probability of the remainder of call classes.

In the third simulation experiment the network is offered traffic with the same structure as in the earlier experiment. In this case the main assumption is that three classes of calls (previously it was only one class of calls) can use overflow links that are located in the first stage. Such an approach leads to a significant decrease in the internal blocking for all traffic classes—for the two classes of elastic traffic, a decrease in blocking is effected both by the operation of the threshold mechanism and the overflow links. By comparing the obtained results it can be concluded that the application of threshold mechanisms alongside overflow links for all classes of calls results in a significant decrease in the internal blocking probability. An application of overflow links to a selected class of real-time traffic only (the second experiment) is followed by the highest decrease in the internal blocking probability for this particular class of calls. The application of overflow links for all call classes also results in a decrease in the internal blocking probability for the classes of elastic traffic and, in consequence, leads to the most effective operation of the switching network for the load corresponding to the range of the real load of the

network. Under these circumstances, the switching network under consideration can be treated as a quasi-non-blocking network.

4 Conclusions

This article presents the results of a simulation of a three-stage switching network that services multi-service traffic with the application of overflow links and threshold mechanisms. The results of the simulation confirm the validity and feasibility of the applied methods. The application of overflow links, dedicated for a number of selected traffic classes, allows the internal blocking phenomenon to be virtually eliminated. In practice, this solution can be applicable for real-time traffic classes. Furthermore, the application of the threshold mechanism makes it possible to significantly decrease losses in elastic traffic streams. The article considers also a possibility of a simultaneous application of the overflow mechanism for all classes of calls. The study confirms high effectiveness of the switching network with such adopted assumptions. Under these assumptions, multi-service switching network can be treated as a quasi-non-blocking network.

References

1. Głąbowski, M.: Modeling of Multi-rate Systems with BPP Traffic Streams. Poznan University of Technology Press, Poznan (2009)
2. Bonald, T., Roberts, J.: Internet and the Erlang formula. ACM Comput. Commun. Rev. **42**(1), 23–30 (2012)
3. Zwierzykowski, P.: Modeling of Traffic Control Mechanisms in Multi-service Mobile Networks. Poznan University of Technology Press, Poznan (2014)
4. Stasiak, M., Głąbowski, M., Wiśniewski, A., Zwierzykowski, P.: Modeling and Dimensioning of Mobile Networks: From GSM to LTE. Wiley (2011)
5. Kaufman, J.S.: Blocking shared with retrials in a completely resource environment. J. Perform. Eval. **15**, 99–113 (1992)
6. Moscholios, I., Logothetis, M., Kokkinakis, G.: Connection dependent threshold model: a generalization of the Erlang multiple rate loss model. J. Perform. Eval. **48**, 177–200 (2002)
7. Vassilakis, V., Moscholios, I., Logothetis, M.: The extended connection-dependent threshold model for elastic and adaptive traffic. In: Proceeding of the 5th International Symposium on Communication Systems, Networks and Digital Signal Processing, Patras, Greece, pp. 42–45 (2006)
8. Głąbowski, M., Sobieraj, M.: Point-to-group blocking probability in switching networks with threshold mechanisms. In: Proceeding of Fifth Advanced International Conference on Telecommunications, IEEE Computer Society, Venezia, pp. 95–100 (2009)
9. Głąbowski, M., Sobieraj, M.: Multi-service switching networks with resource management mechanisms. Mediterr. J. Comput. Netw. **7**(4), 274–282 (2011)
10. Sobieraj, M.: Modeling of switching networks with threshold mechanisms and multi-service traffic sources. PhD thesis, Department of Electronics and Telecommunications, Poznan University of Technology (2014) in Polish
11. Fortet, R. (ed.): Calcul d'organes, Systeme Pentaconta. L.M.T, Paris (1961)

12. Inose, H., Saito, T., Kato, M.: Three-stage time-division switching junctor as alternate route. Electron. Lett. **2**(5), 78–84 (1966)
13. Katzschner, L., Lorcher, W., Weisschuh, H.: On an experimental local PCM switching network. In: Proceeding of International Seminar on Integrated System for Speech, Video and Data Communication, Zurich, pp. 61–68 (1972)
14. Ershova, E., Ershov, V.: Cifrowyje sistiemy raspriedielenia informacji, Radio i Swiaz (1983) in Russian
15. Stasiak, M.D., Zwierzykowski, P.: Performance study in multi-rate switching networks with additional inter-stage links. In: Proceeding of the Seventh Advanced International Conference on Telecommunications (AICT 2011), St. Marteen, Holland (2011)
16. Głąbowski, M., Stasiak, M.D.: Switching networks with overflow links and point-to-point selection [in]: information systems architecture and technology. In: Networks Design and Analysis, pp. 149–159. Wroclaw University of Technology Press (2012)
17. Głąbowski, M., Stasiak, M.D.: Recurrent method for blocking probability calculation in switching networks with overflow links. J. Telecommun. Inf. Technol. **1**, 56–64 (2013)
18. Głąbowski, M., Stasiak, M.D.: Modelling of multiservice switching networks with overflow links for any traffic class. IET Circ. Devices Syst. **8**(5), 358–366 (2014)
19. Roberts, J., Mocci, V., Virtamo, I. (eds.): Broadband Network Teletraffic, Final Report of Action COST 242. Commission of the European Communities. Springer, Berlin (1996)
20. Tyszer, J.: Object-oriented Computer Simulation of Discrete-Event Systems. Kluwer (1999)

Game-Theoretical Approach to Capacity Allocation in Self-managed Virtual Networks

Dariusz Gąsior

Abstract Constantly developing new network services impose changes in net-work architectures. Different applications require different network concepts. Moreover, they should work simultaneously. The necessity of co-existence variety network solutions makes the idea of virtualization vital for contemporary distributed systems. On the other hand, this leads to the rising complexity of network administrating procedures, so the operating costs rapidly grow. The remedy for these problems is automating the administrating methods with the self-management apparatus. In this paper, the capacity allocation problem was modeled with Game Theory approach and the algorithm solving the game (finding Nash Equilibrium) was introduced. Finally, some properties of the game are also discussed.

Keywords Algorithmic game theory · Virtualization · Optimization · Self-management

1 Introduction

The virtualization and the self-managing are two crucial concepts in contemporary networks [1]. The idea of virtual networks enables coexistence of different network architectures (especially New Generation Network architectures like quality of service networks [2, 3] or content-aware network [4, 5]) in one physical infrastructure without interfering each other. Thus, the appropriate network may be created in response to the particular application requirements. For example at the same time there may exist delay-sensitive network for eHealth functions and high throughput network for multimedia purposes. The organization of virtual networks is a multilevel hierarchical structure, which is depicted in Fig. 1. There are many implementation of the network virtualization concept (e.g. links virtualization

D. Gąsior (✉)
Department of Informatics, Wroclaw University of Technology,
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
e-mail: Dariusz.Gasior@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_12

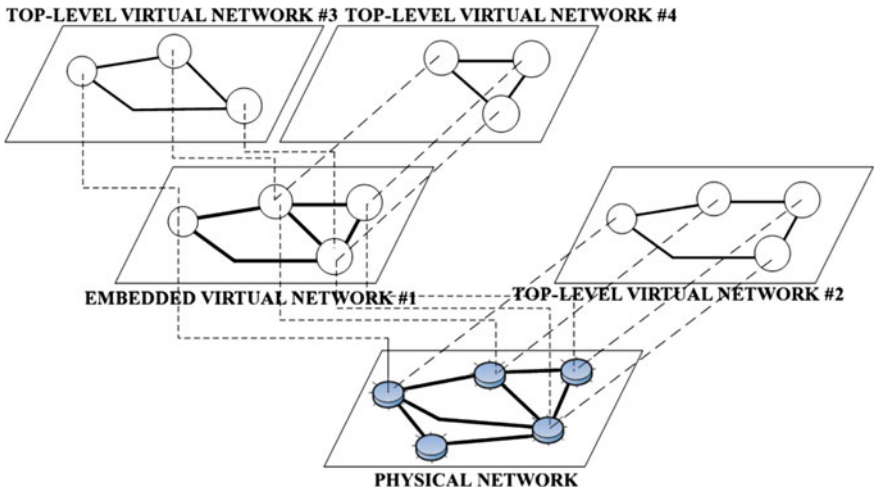

Fig. 1 The concept of virtualization of computer networks

techniques like IP-in-IP, GRE, etc. [6] and node devices enabling virtualization [7]). It makes the resource management to be even more complex task than usually. Thereat the idea of self-management seems to be promising approach. Nevertheless, the efficient algorithms for resource allocation are still missing [8].

The self-managing networks derive from the autonomic networking idea, which was introduced by IBM [9]. A similar concept underlies the self-organizing networks (SON) [10]. This approach was successfully applied for traditional, physical networks, especially for the mobile networks [11]. In [12] authors indicates that existing protocols like well known Transmission Control Protocol (TCP) or Open Shortest Path First (OSPF) may be viewed as the beginnings of autonomic networking.

A survey of the current results in autonomic network management may be found in [13] and [14]. The most common approaches adapt control theoretic approach [15], biological inspired mechanisms [16] and game theory [17].

In this paper, the efficient self-managing algorithm for capacity allocation in virtual networks environment with one level virtualization (i.e. without embedded virtual networks) basing on the game theory is presented.

2 Capacity Allocation Problem in Virtual Networks

Let us assume that there is one physical network and K traffic carrying virtual networks. The physical network consists of L links. Without loss of generality, it is assumed that every virtual network has also L virtual links. Each virtual link corresponds to appropriate physical link. The lth physical link has a capacity Cl,

while the corresponding lth virtual link in kth virtual network has a capacity ckl. There are Rk traffic flows in kth virtual network. The routing matrix for traffic flows is given by binary matrix arkl. Let us denote by xrkl capacity allocated to the rth traffic flow in kth virtual network on lth virtual link. So, the transmission rate of such flow is given by $\bar{x}_{rk} = \min_{l:a_{rkl}=1} x_{rkl}$. The higher is the transmission rate, the more money the flow's owner is willing to pay. This dependence is clearly explained in terms of the network utility [18, 19]. So, we assume that there is given the utility function frk for each transmission flow which reflect the possible income for serving the particular transmission with the given rate $\bar{x}_{rk}$. Usually, the utility function is given in the form of:

$$f(\bar{x}_{rk}) = \begin{cases} w_{rk}(1-\alpha)^{-1}(\bar{x}_{rk})^{(1-\alpha)} & \alpha > 0, \alpha \neq 1, \\ w_{rk} \ln \bar{x}_{rk} & \alpha = 1, \end{cases}$$

where w_{rk} is a parameter characterizing the value of the transmission flow.

Once we assume that all networks are the property of the same owner, adapting the network utility maximization framework, we may formulate the following capacity allocation problem:Given:

K, L, C_l, a_{rkl}, f_{rk}

Find:

$$(x^*, c^*) = \arg\max \sum_{k=1}^{K} \sum_{r=1}^{R_k} f(\bar{x}_{rk}) \tag{1}$$

such that:

$$\forall_k \forall_l \quad \sum_{r=1}^{R_k} x_{rkl} \leq c_{kl},$$

$$\forall_l \quad \sum_{k=1}^{K} c_{kl} \leq C_l,$$

$$\bar{x}_{rk} = \min_{l:a_{rkl}=1} x_{rkl},$$

$$\forall_k \forall_l \quad c_{kl} \geq 0,$$

$$\forall_k \forall_l \forall_r \quad x_{rkl} \geq 0,$$

where $x = [x_{rkl}]_{r=1,2,\ldots,R_k;k=1,2,\ldots,K;l=1,2,\ldots,L}$ and $c = [c_{kl}]_{k=1,2,\ldots,K;l=1,2,\ldots,L}$.

There are some algorithms, which enable finding optimal solution for such an allocation problem in virtual networks, e.g. in [20, 21] even in presence of uncertainty [22]. However, the proposed methods require collaboration and an information exchange between the networks. However, the key feature of the virtualization

concept is that every virtual network has a different owner. In this paper, such a situation is considered. Moreover, it is assumed that each network device in every network should work independently enabling self-managing feature.

3 Game Definition

The problem of capacity allocation in self-managed virtual networks environment may be seen as the game. The networks devices (usually corresponding to the network links) may be seen as the players. Their goal is allocate their resources (e.g. links' capacities), so their local profit is maximized. However, their income depends on the other devices' resource allocations. Thus, in this paper, the appropriate game is proposed to model such a situation. More precisely, two versions of the game are proposed. First one describes the general case for any network structure, while the second one is limited to the single link network topology.

3.1 General Case

Let us introduce the following game, which is referred to as **GVNG** (General Virtual Network Game) in this paper. We assume that every link in every network is a player. So, we have $(K + 1) * L$ players (one physical network and K virtual networks have L links each). For convenience, it is assumed that the physical network is indexed with zero.

For lth physical link player, the permissible pure strategy is any vector $s_{0l} = \bar{c}_l \in S_{0l}$, $\bar{c}_l = [\bar{c}_{kl}]_{k=1,2,\ldots,K}$ where $\bar{c}_{kl}$ denotes the maximal amount of lth physical link's capacity which kth virtual network is permitted to use and the set of feasible strategies is defined as follows:

$$S_{0l} = \left\{ s_{0l} : \sum_{k=1}^{K} \bar{c}_{kl} \leq C_l \wedge \forall_k \bar{c}_{kl} \geq 0 \right\}.$$

For lth virtual link of kth virtual network player, the permissible pure strategy is given by the vector $s_{kl} = [\underline{c}_{kl}, \gamma_{kl}] \in S_{kl}$, $\gamma_{kl} = [\gamma_{rkl}]_{r=1,2,\ldots,R_k}$, where γ_{rkl} denotes the fraction of lth virtual link's capacity that is allocated to the rth traffic flow in the kth virtual network, $\underline{c}_{kl}$ is the maximal amount of lth link's capacity which kth virtual network is willing to use and the set of feasible strategies is defined as follows:

$$S_{kl} = \left\{ s_{kl} : \sum_{r=1}^{R_k} a_{rkl}\gamma_{rkl} \leq 1 \wedge \gamma_{rkl} \in [0,1] \wedge \underline{c}_{kl} \geq 0 \right\}$$

The payoff of the physical link player is given as follows:

$$Q_{0l} = \sum_{k=1}^{K} g_{kl} \min\{\underline{c}_{kl}, \bar{c}_{kl}\}$$

where g_{kl} is an unit price of the lth physical link capacity. The formula $\min\{\underline{c}_{kl}, \bar{c}_{kl}\}$ calculates the actual amount of capacity of lth physical link that is leased to the kth virtual network.

The payoff of the lth virtual link of kth virtual network is given by:

$$Q_{kl} = \sum_{r=1}^{R_k} a_{rkl} b_{rk} f_{rk}(\bar{x}_{rk}) - g_{kl} \min\{\underline{c}_{kl}, \bar{c}_{kl}\}$$

where $b_{rk} = \left(\sum_{l=1}^{L} a_{rkl}\right)^{-1}$ and $\bar{x}_{rk} = \min_{l:a_{rkl}=1} (\gamma_{rkl} \min\{\underline{c}_{kl}, \bar{c}_{kl}\})$.

One must notice that for such defined payoff functions, the social welfare is given by:

$$SW = \sum_{k=0}^{K} \sum_{l=1}^{L} Q_{kl} = \sum_{k=1}^{K} g_{kl} \min\{\underline{c}_{kl}, \bar{c}_{kl}\} + \\ + \sum_{k=1}^{K} \sum_{l=1}^{L} \left(\sum_{r=1}^{R_k} a_{rkl} b_{rk} f_{rk}(\bar{x}_{rk}) - g_{kl} \min\{\underline{c}_{kl}, \bar{c}_{kl}\} \right) = \sum_{k=1}^{K} \sum_{l=1}^{L} f_{rk}(\bar{x}_{rk})$$

which is identical to the objective in (1).

3.2 Special Case

Let us consider special case of GVNG, when there is only one link being virtualized, i.e. L = 1. This case is referred to as SLVNG (Single Link Virtual Networks Game) in this paper.

For such a case, it is easy to obtain a strategy, which gives the greatest value of social welfare, being also the solution of (1).

The optimal transmission rates in virtual networks are given by:

$$\bar{x}^*_{rk} = \begin{cases} \left(\frac{w_{rk}}{g_{k1}}\right)^{\frac{1}{\alpha}} c^*_{k1} \left(\sum_{q=1}^{R_k} \left(\left(\frac{w_{qk}}{g_{k1}}\right)^{\frac{1}{\alpha}} \right) \right)^{-1} & c^*_{k1} \leq \sum_{q=1}^{R_k} \left(\left(\frac{w_{qk}}{g_{k1}}\right)^{\frac{1}{\alpha}} \right), \\ \left(\frac{w_{rk}}{g_{k1}}\right)^{\frac{1}{\alpha}} & \text{otherwise,} \end{cases}$$

where c^*_{k1} is the optimal capacity of virtual link in kth virtual network.

To find c^*_{k1}, one should solve the following linear programming problem:

$$c^* = [c^*_{11}, c^*_{21}, \ldots, c^*_{K1}] = \arg\max \sum_{k=1}^{K} g_{k1} c_{k1}$$

such that:

$$\forall_k \quad c_{k1} \leq \sum_{r=1}^{R_k} \left(\left(\frac{w_{rk}}{g_{k1}} \right)^{\frac{1}{\alpha}} \right),$$

$$\sum_{k=1}^{K} c_{k1} \leq C_1,$$

$$\forall_k c_{k1} \geq 0$$

.

4 The Algorithms for Pure Nash Equilibrium

The Nash Equilibriums are import strategy profiles of the proposed game. They correspond to such a situation when no player is willing to change their allocation while they cannot increase their incomes in such a way. Regarding the self-managed virtual networks, the most payable algorithm for any device is the one, which finds a strategy in Nash Equilibrium strategy profile.

4.1 Algorithm for Nash Equilibrium of GVNG

Let us consider an auxiliary game in the kth virtual network. Let us assume that kth virtual network has fixed virtual links' capacities. There are L players associated with virtual links. Each player strategy consists in dividing his capacity among all transmission flows traversing corresponding virtual link, i.e. $\hat{s}_{kl} = \gamma_{kl} = [\gamma_{rkl}]_{r=1,2,\ldots,R_k}$. The payoff function is given by $\hat{Q}_{kl} = \sum_{r=1}^{R_k} a_{rkl} b_{rk} f_{rk}(\min_{q:a_{rkq}=1} (\gamma_{rkq} c_{kq}))$. The algorithm for finding Nash Equilibrium, which is referred to as BURA (Bottleneck Utility-based Rate Allocation), was presented in [23].

The algorithm finding Nash Equilibrium of GVNG, which is referred to as **CAVE** (Capacity Allocation in Virtual networks Environment), is as follows:

1. For every *k*th virtual network and for every virtual link: find the capacity allocation x_{rkl} using BURA and assuming that virtual link capacity is given by $c_{kl} = \min\{C_l, \hat{c}_{kl}\}$, where $\hat{c}_{kl} = \sum_{r=1}^{R_k} a_{rkl} (w_{rk}/g_{kl})^{1/\alpha}$.

2. Calculate the following formulas:
   (a) $\Delta_{kl} = \sum_{r=1}^{R_k} a_{rkl} x_{rkl}$ for every virtual link l in every virtual network k,
   (b) $G_k = \sum_{l=1}^{L} g_{kl} \Delta_{kl}$ for every virtual network k,
   (c) $\gamma_{rkl} = \frac{x_{rkl}}{\Delta_{kl}}$ for every transmission flow r in every virtual link l in every virtual network k.
3. Solve the following linear programming problem:

$$(\beta_1^*, \beta_2^*, \ldots, \beta_K^*) = \arg \max_{(\beta_1, \beta_2, \ldots, \beta_K)} \sum_{k=1}^{K} \beta_k G_k$$

subject to:

$$\forall_l \quad \sum_{k=1}^{K} \beta_k \Delta_{kl} \leq C_l,$$

$$\forall_k \quad \beta_k \in [0, 1]$$

4. For every virtual link in every virtual network, calculate $\underline{c}_{kl} = \bar{c}_{kl} = \beta_k^* \Delta_{kl}$.
5. The strategy of lth virtual link in kth virtual network is found in step 2 and 4, while the strategy of lth physical link is calculated in step 4.

4.2 Algorithm for Nash Equilibrium of SLVNG

The CAVE algorithm may be also applied to find Nash Equilibrium of SLVNG. However, since the SLVNG is simpler than GVNG, the easier algorithm may be proposed.

Let us introduce the following algorithm finding the strategy profile for SVNG, which is referred to as **SLAVE** (Single Link capacity Allocation in Virtual networks Environment):

1. Let us assume, that virtual networks are ordered according to the non-increasing values of g_{k1}, i.e. for any two indexes of virtual networks k_1 and k_2, $k_1 < k_2$ if and only if $g_{k_1 1} \geq g_{k_2 1}$.
2. For each virtual network k initialize the following values:
   (a) $\hat{c}_{k1} := \sum_{r=1}^{R_k} (w_{rk}/g_{k1})^{1/\alpha}$
   (b) $\underline{c}_{k1} := 0$

(c) $\bar{c}_{k1} := 0$

(d) for every transmission flow r: $\gamma_{rk1} := (w_{rk}/g_{k1})^{1/\alpha}\hat{c}_k^{-1}$

3. Initialize auxiliary variable: $C' := C_1$
4. $\underline{c}_{k1} := \min\{C', \hat{c}_{k1}\}$.
5. $\bar{c}_{k1} := \underline{c}_{k1}$
6. $C' := C' - \bar{c}_{k1}$
7. $k := k + 1$
8. If $C' = 0$ or $k > K$ then STOP, else go to 4.

5 Game Properties

There are some important properties concerning the game, which reflects the quality of possible game solutions, namely: existence of pure Nash Equilibrium, Price of Anarchy (PoA), Price of Stability (PoS) [24], weak Pareto-optimality, strong Pareto-optimality. In this paper, the properties of GVNG and SLVNG are presented. However, only the sketches of proofs are given instead of accurate reasoning.

5.1 Properties of GVNG

CAVE algorithm finds pure Nash Equilibrium of GVNG. It is clear that no player may selfishly change his strategy in order to increase his payoff. A physical link player cannot increase value of $\min\{\underline{c}_{kl}, \bar{c}_{kl}\}$ for any k by changing his strategy (i.e. by changing any value of vector $\bar{c}_l$). Thus, physical link players cannot increase their payoffs. Strategies of virtual links' players are calculated with BURA algorithm, which is proved to find Nash Equilibrium (see [23]).

However, one may easily notice, that for GVNG, the strategy profile containing $\underline{c}_{kl} = 0$ and $\bar{c}_{kl} = 0$ for every k and l also constitutes Nash Equilibrium. In such a case, the social welfare value is equal to 0 (the worst possible) and the Price of Anarchy (PoA) is unbounded.

5.2 Properties of SLVNG

SLAVE algorithm finds pure Nash Equilibrium of SLVNG. The strategy profile found with SLAVE is identical to the one found with CAVE algorithm, since the described procedure allows to solve the linear programming problem given in Step 3 of CAVE algorithm.

It is obvious that SLVNG has the same properties that GVNG, since SLVNG is a special case of GVNG. Thus, the strategy profile satisfying $\underline{c}_{k1} = 0$ and $\bar{c}_{k1} = 0$ for every k also constitutes Nash Equilibrium. Concluding, PoA of SLVNG is also unbounded.

It is noteworthy that SLAVE finds the Nash Equilibrium for the SLVNG as well as solves the problem (2). So, the SLAVE finds Nash Equilibrium, which maximize social welfare. Immediately, one may conclude that Price of Stability is 1.

SLAVE finds the weak Pareto optimal Nash Equilibrium. It seems obvious since one cannot change the strategy profile, so that every player income increases. It is enough to notice that for example physical link player strategy maximizes his payoff despite the virtual links capacity allocations.

Moreover, SLAVE finds the strong Pareto optimal Nash Equilibrium. This property may be justified because the strategy profile obtained with SLAVE algorithm consists of the results of maximization tasks for every player.

6 Final Remarks and Future Works

In this paper, two variants of the game modeling capacity allocation problem in self-managed virtual networks environment are considered. The algorithms finding pure Nash Equilibriums are given. The game properties are discussed.

Obviously, elaborated solutions need to be further investigated and compared with other approaches to the capacity allocation in virtual network problem.

It is also worth considering taking into account more than one level of virtualization, i.e. incorporate embedded virtual networks. It is also promising to examine cases, when the top-level virtual networks are quality of service networks or content aware networks (CAN). While some results concerning self-managed CAN networks have been already obtained [25], especially with game-theoretical approach [26].

References

1. Chowdhury, N., Boutaba, R.: A survey of network virtualization. Comput. Netw. **54**(5), 862–876 (2010)
2. Subharthi, P., Jianli, P., Raj, J.: Architectures for the future networks and the next generation Internet: a survey. Comput. Commun. **34**(1), 2–42 (2011)
3. Lee, C.S., Knight, D.: Realization of the next-generation network. IEEE Commun. Mag. **43** (10), 34–41 (2005)
4. Frank, B., Poese, I., Smaragdakis, G., Uhlig, S., Feldmann, A.: Content-aware traffic engineering. ACM SIGMETRICS Perform. Eval. Rev. **40**(1), 413–414 (2012)
5. Świątek, P., Juszczyszyn, K., Brzostowski, K., Drapała, J., Grzech, A.: Supporting content, context and user awareness in Future Internet applications. In: The Future Internet, pp. 154–165. Springer Berlin Heidelberg (2012)

6. Papadimitriou, P., Maennel, O., Greenhalgh, A., Feldmann, A., Mathy, L.: Implementing network virtualization for a Future Internet. In: Proceeding of 20th ITC Specialist Seminar (2009)
7. Wang, Y., Keller, E., Biskeborn, B., van der Merwe, J., Rexford, J.: Virtual routers on the move: live router migration as a network-management primitive. SIGCOMM Comput. Commun. **38**(4), 231–242 (2008)
8. Chowdhury, N., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Commun. Mag. **47**(7), 20–26 (2009)
9. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Comput. **36**(1), 41–50 (2003)
10. Bettstetter, C., Prehofer, C.: Self-organisation in communication networks: principles and design paradigms. IEEE Commun. Mag. **43**(7), 78–85 (2005)
11. Hu, H., Zhang, J., Zheng, X., Yang, Y., Wu, P.: Self-configuration and self-optimization for LTE networks. IEEE Commun. Mag. **48**(2), 94–100 (2010)
12. Mortier, R., Kiciman, E.: Autonomic network management: some pragmatic considerations. In: Proceedings of the ACM SIGCOMM Workshop on Internet Network Management, pp. 89–93 (2006)
13. Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. ACM Trans. Auton. Adapt. Syst. **1**(2), 223–259 (2006)
14. Agoulmine, N.: Autonomic Network Management Principles. From Components to Applications. Elsevier (2011)
15. Diao, Y., Hellerstein, J.L., Parekh, S., Griffith, R., Kaiser, G., Phung, D.: Self-managing systems: a control theory foundation. In: Proceedings of 12th IEEE International Conference and Workshops on the Engineering of Computer-based Systems, pp. 441–448 (2005)
16. Balasubramaniam, S., Botvich, D., Donnelly, W., Foghlú, M.Ó., Strassner, J.: Biologically inspired self-governance and self-organisation for autonomic networks. In: Proceedings of the 1st International Conference on Bio-inspired Models of Network, Information and Computing Systems (2006)
17. Kelly, F.P., Maulloo, A.K., Tan, D.K.H.: Rate control for communication networks: shadow prices, proportional fairness and stability. J. Oper. Res. Soc. **49**(3), 237–252 (1998)
18. MacKenzie, A.B., Wicker, S.B.: Game theory and the design of self-configuring, adaptive wireless networks. IEEE Commun. Mag. **39**(11), 126–131 (2001)
19. Gasior, D.: QoS rate allocation in computer networks under uncertainty. Kybernetes **37**(5), 693–712 (2008)
20. He, J., Zhang-Shen, R., Li, Y., Lee, C.Y., Rexford, J., Chiang, M.: DaVinci: dynamically adaptive virtual networks for a customized internet. In: Proceedings of the 2008 ACM CONEXT Conference (2008)
21. Drwal, M., Gasior, D.: Utility-based rate control and capacity allocation in virtual networks. In: Proceedings of the 1st European Teletraffic Seminar (2011)
22. Gasior, D.: Capacity allocation in multilevel virtual networks under uncertainty. In: XVth International IEEE Telecommunications Network Strategy and Planning Symposium (2012)
23. Gasior, D., Drwal, M.: Pareto-optimal Nash equilibrium in capacity allocation game for self-managed networks. Comput. Netw. **57**(14), 2817–2832 (2013)
24. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: STACS 99. Springer Berlin Heidelberg (1999)
25. Gasior, D., Drwal, M.: Decentralized algorithm for joint data placement and rate allocation in content-aware networks. In: Computer Networks. Springer Berlin Heidelberg (2012)
26. Gasior, D., Drwal, M.: Caching and capacity allocation game in self-managed content provider networks. In: 16th International IEEE Telecommunications Network Strategy and Planning Symposium (2014)

Distributed Algorithm for Text Documents Clustering Based on k-Means Approach

Martin Sarnovsky and Noema Carnoka

Abstract The presented paper describes the design and implementation of distributed k-means clustering algorithm for text documents analysis. Motivation for the research effort presented in this paper is to propose a distributed approach based on current in-memory distributed computing technologies. We have used our Jbowl java text mining library and GridGain as a framework for distributed computing. Using these technologies we have designed and implemented k-means distributed clustering algorithm in two modifications and performed the experiments on the standard text data collections. Experiments were conducted in two testing environments—a distributed computing infrastructure and on a multi-core server.

Keywords Clustering · Text mining · k-Means · Distributed computing

1 Introduction

In general, clustering analysis belongs to a group of unsupervised learning tasks, in which objects are grouped into a set of clusters based on their similarity. The main objective is to create clusters of similar objects based on input data, which doesn't include any information about the classes [1]. A cluster represents the group of objects similar to each other, but different as much as possible from the entities in the other clusters.

There are several types of clustering methods, which are nowadays heavily influenced by the increasing amounts of data processed [2], especially when dealing with the textual data. Clustering of texts is a fully automated process which divides a set of documents into groups. If the content of the documents is used as a means

M. Sarnovsky (✉) · N. Carnoka
Department of Cybernetics and Artificial Intelligence, Faculty of Electrotechnics and Informatics, Technical University Kosice, Letna 9/a, 04001 Kosice, Slovakia
e-mail: martin.sarnovsky@tuke.sk

N. Carnoka
e-mail: noema.carnoka@tuke.sk

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_13

of differentiating between them, different groups then can correspond to different themes appearing within this collection of documents. Documents within one particular cluster are similar, speaking of their content, so clustering of texts can be viewed as a method that can be used to detect the content of the document collections and identify the possible topics. Each of the clusters contains the documents, most related between each other, but on the other hand, as much different from the documents in other clusters as possible. From the machine learning perspective, clustering methods belong to a set of unsupervised training approaches. Training of the models are usually automatic, without a feedback provided by user's input which leaves the particular algorithm to find the patterns within the data. Discovered clusters are usually disjoint.

Centroid-based clustering methods are popular and frequently used group of methods. In general, centroid-based methods use centroid as the entity representing the particular cluster. Most popular centroid-based methods include k-means and k-medoids. K-means belongs to the most popular clustering tools being applied in various industrial and scientific scenarios [2].

Input objects are assigned to clusters, the number of which is known in advance. Each of these clusters are represented by the centroid (average mean of the cluster). Standard version of the k-means algorithm usually consists of two steps. The first one represents the initialization—a random selection of the k objects which represent k clusters. Consequently, the objects are assigned to the nearest (according to the Euclidean distance) cluster. Then, in the next step (called update step), when all objects are assigned to the clusters, an algorithm calculates the new mean of created clusters to be the centroids of the new clusters. This is repeated until the centroid positions no longer change.

The main benefit of k-means algorithm is its simplicity. K-means, however, has some disadvantages, including, for example:

- the result is strongly dependent on the initial centroids selection,
- calculated local optimum can differ from a global optimum,
- the process is sensitive to the points outside of all centers etc.

2 Overview of Existing Distributed Approaches to k-Means Clustering

Several approaches to distributed k-means algorithm implemented using various technologies already exists. Our main objective was to inspire ourselves by already investigated approaches and to design a specific one for the text documents clustering based on these experiences.

In [3] authors describe a parallel k-means algorithm which implemented using MPI (Message Parsing Interface). In this approach, each of the processes that run in parallel receive a list of all centroids (initial centroid positions are calculated

on the root). Individual processes calculate distance vectors of each of the local centroids and then store calculations. In the next step, they collect individual vectors totals for the respective cluster and calculate an average of the vectors for the corresponding clusters. In the next step each process assign data to the nearest centroid and calculate a local mean square error (MSE). Then, in the update set, centroid positions are re-calculated locally and local MSE is computed for each parallel process. DFEKM [4] is another approach using parallel or distributed programming techniques, based on the Fast and Exact K-means Clustering (FEKM). This distributed approach is suitable for clustering of the distributed data on the loosely coupled machines. Mahout (a machine learning library built on top of the Hadoop) [5] uses a custom implementation of k-means clustering. In this case, the process of model building consists of three phases: the initial phase, which splits the dataset into HDFS blocks, the second one is the mapping phase, where distances between the data and the centroids are computed and compared to samples with the closest centroid. Then the data are assigned to specific clusters. The last phase is the reduce phase which recalculates the centroid points using average coordinates of all points within the cluster.

3 Design and Implementation of the Distributed Clustering Algorithm Based on k-Means

This section describes the design and implementation of distributed k-means algorithm for text analysis purposes. Methods of text analysis nowadays can have higher requirements on the processing capabilities and therefore can be time consuming, especially in the era of big data when dealing with the large data collections. Therefore leveraging of the newly emerged technologies for distributed computing can bring significant benefits, if utilized properly. Utilization of these methods enables us to perform various text-mining tasks such as text classification and text clustering on the distributed computing infrastructures. We briefly describe the technologies used the implementation and the main features of designed approach.

3.1 Technologies

Our solution is implemented in Java and based on two core technologies. We used the Jbowl library for pre-processing of the data, indexing of the data. We also have used its k-means implementation as the basis for distributed version. On the other hand, GridGain middleware was used to provide framework for distributed computing.

Jbowl (Java Bag-of-words Library)[1] is an original software system developed in Java to support information retrieval and text mining [6]. The system is being developed as open and provides an easy extensible, modular framework for pre-processing, indexing and further exploration of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models. Jbowl is a Java library, which contains methods for preprocessing, classification, clustering (including k-means algorithm) and several evaluation techniques. It provides a set of classes and interfaces that enable integration of various classifiers and clustering algorithms. Jbowl distinguishes between clustering algorithms (GHSOM, k-means) and clustering models (centroid-based clustering models, etc.) [7]. We used implementation of k-means algorithm.

GridGain[2] is the middleware based on Java platform, used to develop distributed applications. It supports development of scalable data-intensive and high-performance distributed applications [8]. GridGain is independent from infrastructure and platform which allows applications to be deployed on various types of infrastructures. GridGain provides native support for Java, Scala and Groovy languages. Framework consists of two core technologies: Compute grid and In-memory data grid. Compute grid technology provides in-memory MapReduce implementation for handling distribution of process logic. In-memory data grid represents the capability to parallelize the data storage by storing partitioned data in memory closer to the application. GridGain is more suitable for real-time data processing utilizing the in-memory computation model. We already have experience in using the GridGain programming model for text classification [9], formal concept analysis [10, 11], information retrieval tasks [12], text clustering [13] and developed the complex information system for text mining purposes using this framework [14].

3.2 *Distributed Model-Building*

Proposed distributed solution includes two main components: the root node and other involved computing resources—worker nodes. The root node is a workstation which manages and governs the entire process. It performs pre-processing and indexing of the dataset and divides the model building process into subtasks. Master node collects information about the actual state of the computing infrastructure. According to the number of available worker nodes, it calculates the number of subtasks, splits the data and distributes the subtasks and data to the worker nodes. Partial clustering models are created on worker nodes (including the root node). Individual partial models are then collected by the root node and combined into one final clustering model.

[1] http://sourceforge.net/projects/jbowl/.

[2] http://www.gridgain.com/.

Our distributed version of k-means clustering algorithm is based on methods presented in [15] and [16]. Our approach separates the process of creation of k clusters among the available computing resources. Creation of the particular clusters (computation and re-calculation of the particular centroid) represent the subtask, so k separate subtasks are created when building a model with k clusters. Subtasks are distributed within the environment are performs locally on the assigned node and on the assigned data. We proposed two different modifications of our approach. The difference is in initialization phase and in splitting the data for subtasks.

In the first solution we decided to divide the dataset into the subsets. Clusters are initialized on the subsets and subtasks are scheduled. There are as many data subsets created as available nodes in the infrastructure. Data subsets are then distributed to assigned worker nodes with assigned subtasks to be performed on the data subset. Subtasks performed on these nodes correspond to computation of k-centroids. Centroids are in this case initialized globally on the master node and recalculated by assigned node on particular subset of data. When the locally computed centroids do not change their positions (or specified level of iterations has been reached), partial models are retrieved to master node to merge into the final model.

The second modification of our solution, the dataset is not split into the subsets. The main difference is, that the root node computes the initial cluster centroids globally on the complete dataset and then distributes the subtasks (corresponding to the particular centroid calculation) to the worker nodes. Each worker node have access to the whole dataset in order to re-compute the centroid's position. The reason for this modification was to enable the subtasks to cover entire dataset and create more general clusters describing the dataset. Similarly as in the first version, worker nodes return the partial models to the master node which then constructs the final one. The aim is to compare these approaches from the time processing perspective, but also from the perspective of models themselves.

4 Experiments and Evaluation

The main goal of the experimental was to show how the division of the algorithm into partial subtasks reduce a time-consuming creation of k-means model. Another objective was to compare the two implemented approaches on the same data and with the same settings.

We have used two datasets for experimental evaluation. The first dataset was Reuters-21578 ModApte consisting of a collection of Reuters articles published in 1987 with mainly economic focus. This dataset is split into the testing and training set and we used the training set which includes a 7769 documents with 22,526 terms. Both subsets contain documents in 90 different categories (). The collection is available publicly in SGML format and for our experimental purposes was transformed into the XML. The second dataset was Medline, a bibliographic

database of the US National Library of Medicine. Dataset contains text documents archived since 1950 and comprises of more than 11 million articles and more than 4800 indexed records. In this work we have used OHSUMED collection, a Medline subset, containing documents published in 1990. Documents in dataset describe the domain of heart diseases (5192 types) and contain 49,580 documents in 90 categories using 171,936 terms. This dataset also has been transformed into the XML format.

Experiments were conducted on the infrastructure deployed in one of our laboratories. The testbed consisted of 10 workstations with Intel Xeon CPU, 3.06 GHz, equipped with 4 GB RAM and 32-bit operating system. We also used the same implementation on multi-processor platform—a 16 core Intel Xeon, 2.93 GHz server with 16 GB RAM attached. In this case, the worker nodes were separate GridGain threads running using each core. We tried to observe how the implementation would work with the shared memory and to compare the cost of data distribution in distributed version.

In general, we performed a series of test on both datasets. We decided to perform a series of test with different values of the k parameter, as this parameter influences the complexity of the model and using the different resources. We wanted to observe the effect of the subtask distribution on different models with different complexity. We repeated each experiment five times and collected the times needed to construct the final model and then averaged the results for the particular test. The experiments of the second modification of proposed algorithm were not successful due to the nature of the test environment. Nodes were required to process the entire Medline dataset which caused a problem due to the size of java virtual machine memory.

The graph depicted on Fig. 1 represents the results of the first implemented solution running on 1–9 worker nodes with settings of k parameter (number of

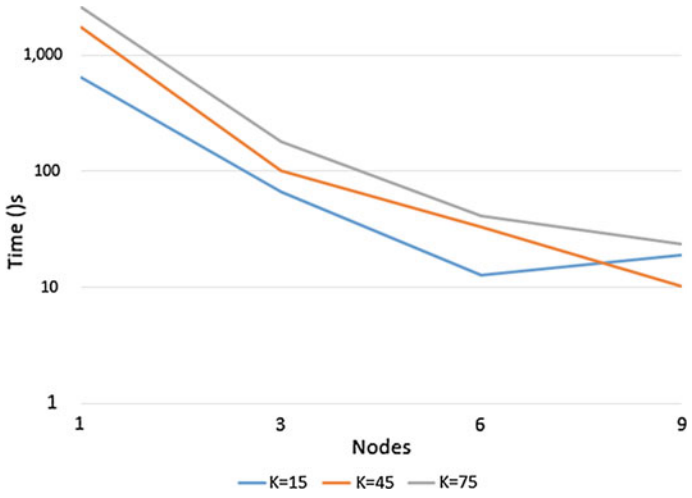

Fig. 1 The first solution, distributed environment, Reuters collection

clusters) to values 15, 45 and 75 (note the logarithmic scale of the time axis) on Reuters dataset. We can observe rapid non-linear reduction of the processing times, which was caused by the reduction of the dataset (nodes processing only subsets) and by the limited amount of computation tasks (only assigned centroids). Adding another nodes to the infrastructure did not bring any benefits (in case of building model with 15 clusters 6 nodes were optimal), as the communication overhead needed to send the data and partial models became costly and lead to increased time to build the final model. Figure 2 represents the results obtained by running the experiments in the parallel mode.

Figs. 3 and 4 depict the results of the second implemented version, also running on 1–9 nodes, using same k parameter values on the same dataset. Differences are visible on single node—caused by the initialization phase. In this case, initialization of the initial centroids was performed globally on the master node, which caused the longer processing times when creating more clusters. The results using the k value set on 75 on 6 nodes were influenced by the unexpected issues with threads processing in one of the total 5 experiments on this particular setup. This caused the averaged time to be higher than expected.

The results achieved on the Medline dataset are summarized in Table 1. Only the first solution is present (running on 1–9 nodes with the same k parameter setting as previous experiments) as the second one did not finished all of the experiments due to the memory issues.

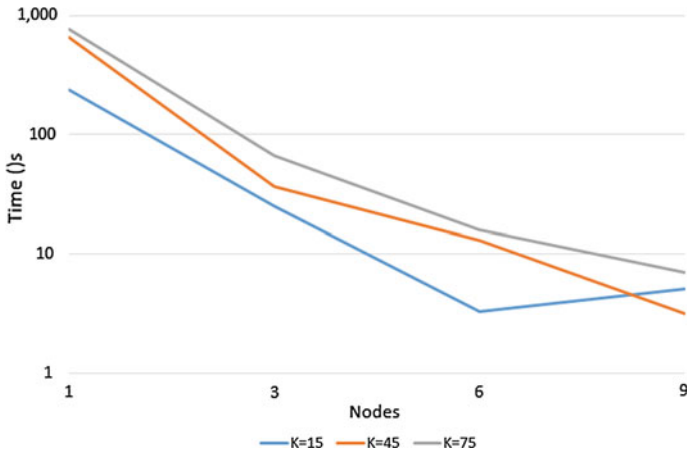

Fig. 2 The first solution, parallel environment, Reuters collection

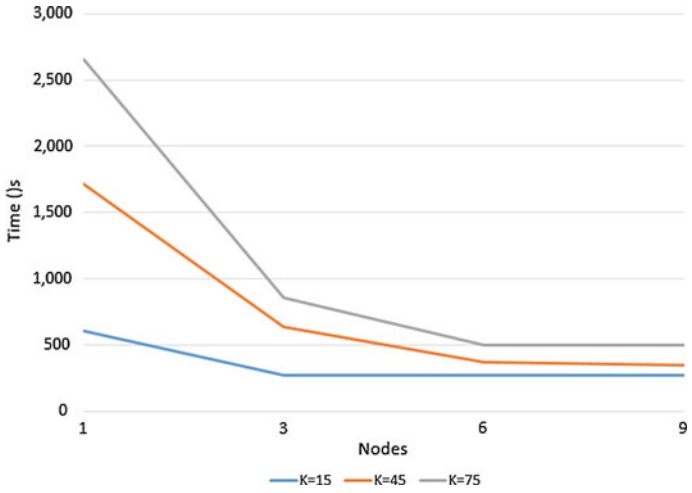

Fig. 3 The second solution, distributed environment, Reuters collection

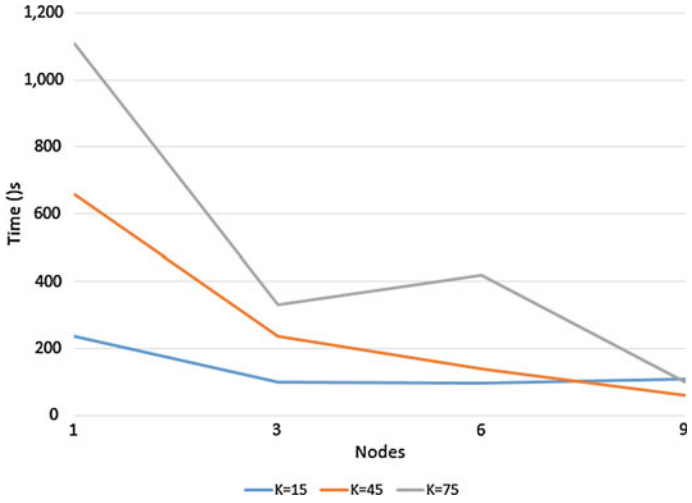

Fig. 4 The second solution, parallel environment, Reuters collection

Table 1 Comparison of the first solution processing Medline collection (times in seconds)

Nodes	K = 15	K = 45	K = 75
1	939,229	2769,491	3891,532
3	71,340	211,774	504,871
6	45,721	65,278	47,951
9	8517	13,439	17,493

5 Conclusion and Future Work

Main objective of the work presented within the paper was to design and implement the k-means algorithm in distributed fashion using the in-memory GridGain framework. In the future, our plan is to build on the previous experiences and results with distributed clustering and classification and enhance the proposed solution with optimization methods to balance the node usage and subtasks complexity. On the other hand, the solution will be integrated into the more complex and coherent information system for text analysis tasks in distributed environments. The system is being developed using above mentioned technologies and provides the graphical visualization tools as well as evaluation techniques. The system is being used in teaching as well as in research tasks and contains user interface for text preprocessing methods, classification and clustering algorithms (trees, SOM clustering, etc.) in distributed form, as well as in research as a platform for both data analysts that uses the existing methods, and researchers providing them a coherent platform for research activities in related area.

Acknowledgements The work presented in this paper was partially supported by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic under grant No. 1/1147/12 (50 %) and as the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF (50 %).

References

1. Paralič, J., Furdík, K., Tutoky, G., Bednár, P., Sarnovský, M., Butka, P., Babič, F.: Text Mining (in Slovak: Dolovanie znalostí z textov). Equilibria, Košice (2010)
2. Andrews, N.O., Fox, E.A.: Recent developments in document clustering. Technical report TR-07-35. Deparment of Computer Science, Virginia Tech (2007)
3. Joshi, M.N.: Parallel K-means algorithm on distributed memory multiprocessors, Project Report, Computer Science Department, University of Minnesota, Twin Cities (2003)
4. Jin, R., Goswami, A., Agrawal, G.: Fast and exact out-of-core and distributed k-means clustering. Knowl. Inf. Syst. **10** (2006)
5. Rui, M.E., Rui, P., Chunming, R.: K-means clustering in the cloud—a Mahout test. In: Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA'11). IEEE Computer Society, Washington (2011)
6. Bednar, P., Butka, P.: Task-based execution engine for JBOWL. In: Proceedings of WIKT, Bratislava, Slovakia, pp. 65–68 (2009)
7. Butka, P., Bednar, P., Babic, F.: Use of task-based text-mining execution engine in support of knowledge creation processes. In: Proceedings of Znalosti, Bratislava, pp. 289–292 (2009)
8. GridGain 3.0.: High performance cloud computing whitepaper (2011). http://www.gridgain.com/media/gridgain_white_paper.pdf
9. Sarnovský, M., Kačur, T.: Cloud-based classification of text documents using the Gridgain platform. In: Proceedings of 7th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2012, Timişoara, Romania (2012)

10. Butka, P., Pocs, J., Pocsova, J.: Distributed version of algorithm for generalized one-sided concept lattices. In: Intelligent Distributed Computing VII Book Series. Studies in Computational Intelligence, vol. 511, pp. 119–129 (2014)
11. Butka, P., Pocs, J., Pocsova, J.: Distributed computation of generalized one-sided concept lattices on sparse data tables. Comput. Inform. **34**(1), 77–98 (2015)
12. Butka, P., Pócsová, J., Pócs, J.: Proposal of the information retrieval system based on the generalized one-sided concept lattices. In: Topics in Intelligent Engineering and Informatics. Applied Computational Intelligence in Engineering and Information Technology, vol. 1. Springer, Berlin (2012)
13. Sarnovsky, M., Ulbrik, Z.: Cloud-based clustering of text documents using the GHSOM algorithm on the GridGain platform. In: SACI (2013), pp. 309–313 (2013)
14. Sarnovský, M., Butka, P.: Cloud computing as a platform for distributed data analysis. In: 7th Workshop on Intelligent and Knowledge Oriented Technologies, WIKT 2012 (2012)
15. Srinath, N.K.: MapReduce design of K-means clustering algorithm. In: Proceedings of International Conference on Information Science and Applications (ICISA) 2013 (2013)
16. Zhang, C., Li, F., Jestes, J.: Efficient parallel kNN joins for large data in MapReduce. In: Proceedings of the 15th International Conference on Extending Database Technology (EDBT'12), pp. 38–49. ACM, New York (2012)

Dictionary as a Service—A Software Tool for Vocabulary Development and Maintenance

Paulina Kwaśnicka, Paweł Stelmach, Krzysztof Juszczyszyn and Grzegorz Kołaczek

Abstract This paper describes an original solution, dedicated for building and maintaining domain vocabularies for service-oriented software systems. The solution (named Dictionary) was implemented as a service and offers a range of tools for efficient and consistent management of domain vocabulary. The vocabulary is then used for modeling and description of service systems, being developed for the client (in this case the final products are optimization tools for transport companies).

Keywords Domain vocabulary · Ontology · Service-oriented systems · SOA

1 Introduction

With the rising numbers of interconnected systems and especially number of cases when heterogeneous systems had to be connected for the first time, a need for some vocabulary standardization has also increased in importance. Even more so, some have discovered that in order to properly manage vocabulary we cannot only focus on words but also on their meaning and how they describe real world.

To answer those needs some solutions have been proposed. First is the ontology, which is one of the more complex tools but also an expressive one, gaining popularity in the past decade. Ontologies appeared first as a branch of philosophy, which focuses on answering questions about entities, relations, events and

P. Kwaśnicka (✉) · P. Stelmach · K. Juszczyszyn · G. Kołaczek
Department of Computer Science, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: paulina.kwasnicka@pwr.edu.pl

P. Stelmach
e-mail: pawel.stelmach@pwr.edu.pl

K. Juszczyszyn
e-mail: krzysztof.juszczyszyn@pwr.edu.pl

G. Kołaczek
e-mail: grzegorz.kolaczek@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_14

processes in real world. The aim of ontology is to classify entities, so that one could answer the question: "What classes of entities are needed for a complete description and explanation of all the goings-on in the universe?" [1].

Ontology in computer science has been used in similar fashion but with a limited scope. Classifications of entities (also called concepts) are created for fragments of the real world so to answer another question: "What classes of entities are necessary for a complete description for a specific domain?". Conceptual schema describes what a particular concept means (concept being a representation of some object in the real world) and what are its relationships with other concepts. Often the description of a concept consists of its name and an identifier conforming to the URI standard (Uniform Resource Identifier). Relationships among concepts describe: what other concepts comprise the particular concept, what other concepts are somehow related to the given concept (but are not part of it), or specifically what other concepts are a generalization or specification of a given concept.

Another popular solution is representing the relationships between words themselves, as proposed in WordNet [2]. WordNet is a lexical database that consists of words in English language. The strength of this particular database is grouping of sets of words in synsets (synonym sets) and using other lexical relations that express how words are related to each other, be that words with broader or narrower meaning, or words that homonyms.

On the other hand, the use of such data structures is not on par with the development standards. The computer software created nowadays often offers ease of access, focus on user's changing functional and non-functional requirements, offering personalization and acknowledges that systems are decentralized and their functions can be made available via services. To meet such demands many new systems are built using the service-oriented approach, as presented in SOA (Service Oriented Architecture) architectural pattern [3].

This approach and consequences of its application to ontologies and dictionaries is the focal point of this paper. Section 2 explains in detail the motivation behind the proposed solution. Section 3 presents key ideas behind the solution that are direct answers to the needs recognized earlier. Section 4 gives insight into implementation of the solution and its application in a business optimization platform built with the SOA approach. Finally, Sect. 5 presents analysis of similar tools, and Sect. 6 concludes the paper with some final remarks.

2 Motivation

Software ontologies are usually designed for specific areas of real world (so called domains), often with a notion that once created they will be used in applications without modifications. However, this might be true in philosophy, but software ontologies should be practical in nature, describing not the invariable reality, but modelling it for use in a specific situation. The requirements for ontology usability might change—this might not be a preferred situation, but it is a typical one

nonetheless. With this in mind, the usual approach of building static ontologies in external, dedicated applications is out-of-date.

There is a need for an approach, enforced by new technologies, to constantly evolve ontologies and use them at the same time.

This motivation was a basis for key concepts behind a new service-based web application, a tool for managing and delivering ontologies and dictionaries, that is simply called in its prototype form "Dictionary", in order not to alienate its typical users, that have little experience with ontologies. At the same time, those inexperienced users are what drives the evolution of the "Dictionary", so that, in contrast to typical ontology editors, it delivers its functions via a user-friendly graphical interface.

3 Key Ideas Behind the Solution

The proposed approach relies on some fundamental assumptions:

- The Dictionary should be relatively easy to use by non-ontology experts (compared to typical tools) and should bring to mind some notions that might be familiar to them in every day's work.
- Having the above in mind, there has be some inherent complexity added to typical dictionary approach, that distinguishes words (which in presented approach are called **terms**) and their meaning (which is represented by ontology concepts and relations among them).
- The Dictionary should be focused on vocabulary evolution and its use at the same time.
- Typical use of the Dictionary isn't in the graphical user interface in which the concepts are defined but in other applications that are external to the Dictionary.
- We should distinguish at least two types of Dictionary's users: ontology creators and consumers: the former use Dictionary's GUI and manage concepts and vocabulary, the latter don't.

The solution that follows these assumptions is a web-based application that is built following the MVC design pattern but also is focused on delivering its functionality via web services, as suggested in the SOA approach.

Contrary to some typical ontology management tools (compared in following sections) this web tool is not focused on ontology itself but a complete set of tools: ranging from ontology management, vocabulary management through ontology and vocabulary evolution process management and is focused on delivery of those structures to less experienced users and, more importantly, external applications.

Some key ideas behind the solution are:

- Separation of logic (business) and realization (data structures): in addition to typical MVC separation, the Dictionary provides additional abstraction layers, hiding the fact that it holds ontologies and vocabularies separately, and allowing for a combined experience for a business user.

- Using familiar concepts: assuming that UML is more popular than OWL (ontology standard), the Dictionary presents concepts using the UML-like approach, dividing possible relationships among concepts into attributes (that are more integral part of a specific concept) and relations (that reflect more external relationships among various concepts). This, however, is dynamically translated into ontology and vocabulary accordingly.
- Using web services to deliver content to other applications: the concepts in the Dictionary are not only browsable via the graphical interface but more importantly can be requested via web services by any application (typically web application) that can execute them; dictionary mechanisms allow for validation of constructs created externally and propose proper Dictionary concepts thus forcing consistency among various client applications.
- Using web services to allow for Dictionary's content evolution: the user that manages the Dictionary and uses its graphical interface isn't typically the one that creates content. More natural environment for content creation is in other applications in which web forms are completed. This way web service based mechanisms of the Dictionary allow for preliminary validation of those forms inputs but also can accept suggestions of new concepts created during this process. However, this is where the Dictionary's expert user verifies those suggestions and after consideration can make changes to ontologies and vocabulary (that dynamically can influence other connected applications).

3.1 Base Entities

The three fundamental entities distinguished by the Dictionary are:

- Terms—terms define the vocabulary; they consist of the name (the string of characters) and an identifier to know apart two terms with the same name. Terms can be perceived as labels for concepts, which names don't mean anything. On this vocabulary layer we distinguish relations of synonymy. One term can be a label for many concepts as well as one concept can have many term labels. In the current implementation, however, this possibility has been limited to verify that it would be controllable by less experienced users (Fig. 1).
- Concept—defines a model for some abstract entity that is a valid conceptualisation in the real world; it itself has no readable name, only an identifier that should be used to define relations with other concepts. Often people create names for concepts to easily manage them, however, this can lead to the misconception that that name is related to the concepts meaning. In fact the only meaning that one could rely on comes from the concepts relationships with other concepts. In the Dictionary to label concepts and identify them by a human user terms are used (Fig. 2) and with them users can convey meaning they attribute to the specific term (however, many of the Dictionary systems will still ignore the name).

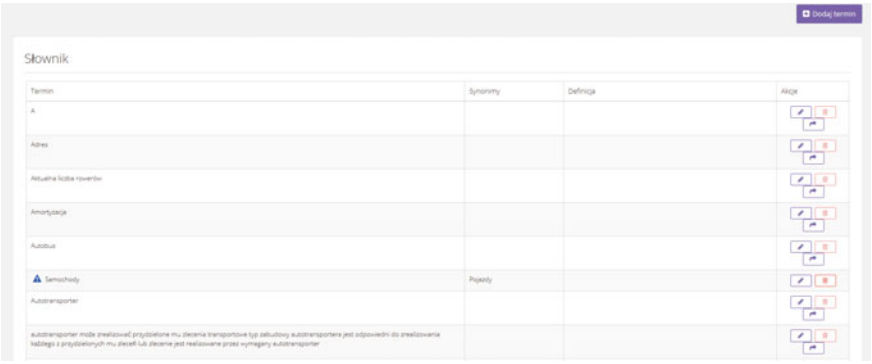

Fig. 1 Term management interface

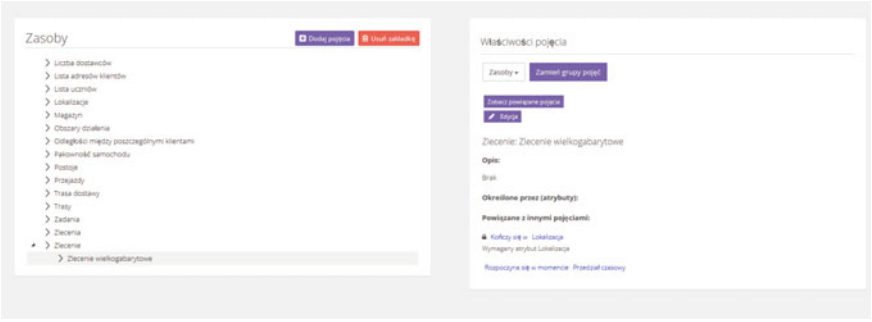

Fig. 2 Concept management interface

- Relations—relations represent typically asymmetric associations among concepts. In dictionary for the sake of the business user, in the Dictionary three kinds of relationships among concepts are highlighted: relations, attributes and generalizations (inheritance), which are treated uniquely by the Dictionary, but that are the base on which users can build their ontologies (Fig. 3). For the definition of what a particular concept really mean the attribute relation is more significant than other relations.

To allow for global identification of all of the above elements, their identifiers have been created in compliance with the Uniform Resource Identifier standard (URI). The identifier always starts with "http://pop.pwr.edu.pl/" and is followed by name of the element type, and domain name "/domain/". Finally, a local identifier based on the used term is generated automatically based on the system clock to guarantee uniqueness.

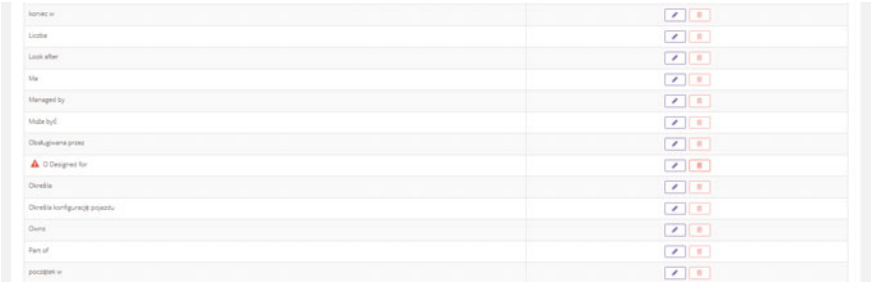

Fig. 3 Relations' management interface

3.2 Dictionary Evolution Process

One of the most important features that distinguish the Dictionary from typical tools like TopBraid Composer or Protégé is the focus on continuous ontology and vocabulary evolution and concurrent use of its resources (Fig. 4).

There is a separate role of a dictionary specialist that uses the graphical user interface and manages both ontology and vocabulary. There is suggestion system implemented for interaction with users that need new or modified concepts. Those users interact with external web applications that in their name use web services to post a suggestion in the Dictionary. Every suggestion has a ticket that can be checked with a designated service. Suggestions can point to existing concepts, proposing their modifications or simply using them to relay a new meaning using known concepts. Finally, the dictionary specialist can automatically use some or all of the suggestions or sometimes manually create or select a concept that in his mind is the best fit for a specific need. This way the Dictionary provides a medium for contextual conversation between interested parties without limiting of their options, instead proposing an unobtrusive automation.

3.3 Dictionary's Concept Hierarchy

When designing a conceptual schema one typically quickly discovers that some concepts are a generalisation or a specification of other concepts. With this in mind we often lean towards classification of things where on top of such hierarchies are most general concepts (on the very top we usually have "Thing" or simply an abstract "Root").

In ontologies often concepts correspond to classes (in the class—individual distinction), which in turn can be interpreted as sets of individuals. In this context, when thinking about concept hierarchy, we can imagine that each individual that would be in a more specific class also would be in a more general one. On the other

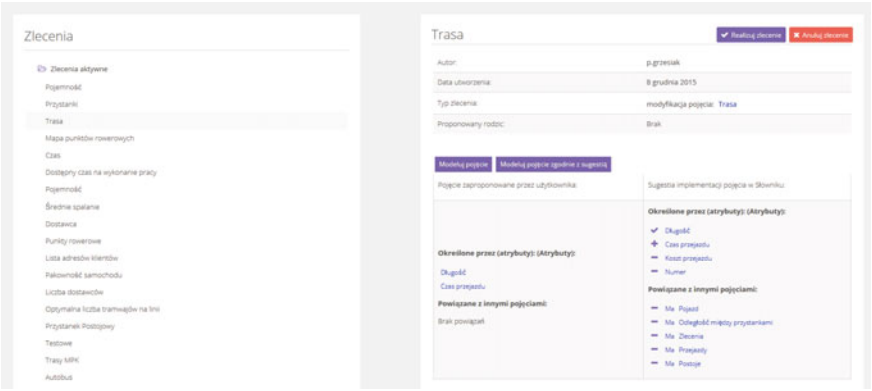

Fig. 4 Dictionary management interface

hand, everything that is true about some individual in the more general class holds true if that individual is also in a more specific class.

In the Dictionary we abstract from classes and individuals and focus on only concepts, however, with concept hierarchy, a rule is enforced, so that each child-concept (more specific) must inherit its parent-concept attributes and relations. Those relations themselves can undergo specification process (the object-concept in the property—attribute or relation—can be replaced with its specification) as long as parent-concept requirements remain fulfilled. Such process, as well as many others, is supported by the Dictionary's GUI, which provides mechanisms for specification of concepts, their attributes and relations as well as validation when knowledge about concepts is modified during the system's evolution.

4 Application

The Dictionary web tool has been created using the ASP.NET MVC 5 framework and uses a dedicated relational database, as well as a prototype RDF and OWL database.

The practical need for this kind of tool first appeared during works on the project in the Wrocław University of Technology "Business process optimization platform in integrated information systems". Currently the Dictionary is part of the distributed web platform POP (as part of the aforementioned project) and delivers its functionality to two web applications: Task Modeller and Decision Support Task Repository. In both applications data from the Dictionary is used to help and model

Decision Support Tasks: typically it helps in defining knowledge resources needed on the input or generated on the output, but its functionality was also extended to help model criterions and constraints used in definition of decision support tasks (mostly optimization tasks). It should be noted, that our Platform supports the versioning, consistency checks and the ongoing development of all dictionaries, with its original user interface and user-support functions, described in this paper.

The key aspect of this approach is knowledge accumulation—when consecutive transport organizations are analysed, it is done according to our original Methodology, which supports the business analysis and guarantees its compliance with the dictionary knowledge already gathered during analysis of the former cases. Hence, any new case is tested for similarity with the previously analysed problems (for which the algorithms and working services were already developed).

Thanks to the use of the Dictionary, platform users build models consistent with the selected transport domain (key application domain of the project), without the need to navigate through its graphical interfaces. This approach prevents explosion of modelling approaches while still allowing for sending suggestions of new concepts to the Dictionary. This incremental evolution of vocabulary is further supported by update mechanisms, when the Task Modeller is informed about changed definitions of concepts it might have used (Fig. 5).

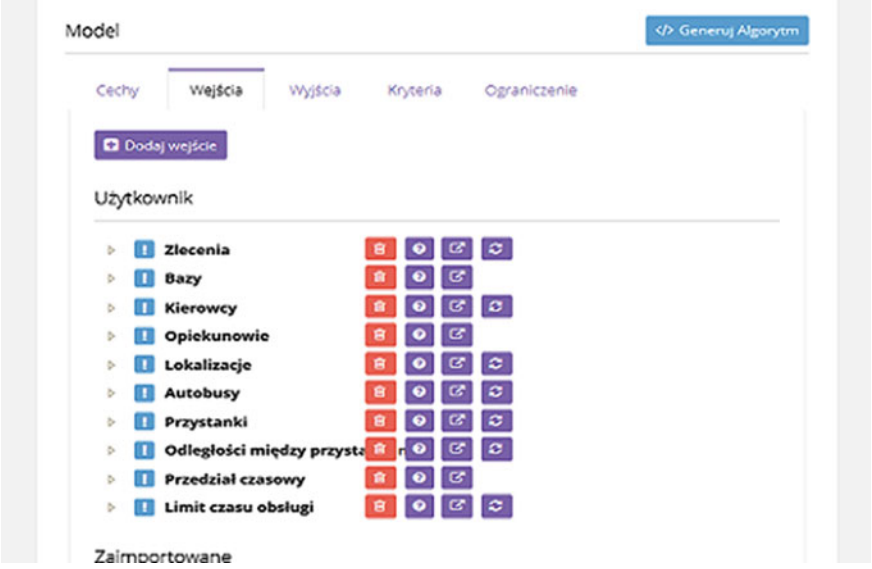

Fig. 5 Task modeler

5 Comparison with Other Tools

5.1 TopBraid Composer

TopBraid Composer is an ontology editor created by TopQuadrant. It uses RDF, RDFS and OWL, which are standard languages for definition of ontologies and resources [4, 5]. Compared to the Dictionary the main difference lies in the modelling process. TopBraid focuses on defining concepts in a specific standard, like a more sophisticated text editor, whereas the Dictionary focuses on the planned use of those concepts, hiding their implementation under a business layer.

Also, in TopBraid the name of the concept is not separated from the concept as in the Dictionary. TopBraid does not distinguish attributes of concepts, connecting them only by general-purpose relations or hierarchy, however, it allows for relations between concepts and non-concepts (via data type properties e.g. numbers), whereas the Dictionary only allows for relations between concepts.

TopBraid Composer has been created as a desktop application, however, the Maestro Edition offers the capability of creating web applications based on the TopBraid Live platform. In comparison, the Dictionary itself is a web application and doesn't need any additional components to work and share its resources.

5.2 Protégé

Protégé is a tool created by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. Similarly to TopBraid it is compliant with RDF and OWL [6, 7]. Its approach to modelling ontologies is also similar to TopBraid, because both take on the role of editors for OWL-type ontologies rather than tools for general ontology and vocabulary management.

Protégé however offers an extension via a SKOS Editor that has a capability to denote concepts similar to each other and those that can be used interchangeably. Yet, using SKOS and the aforementioned extension requires manual linking of those concepts, where this mechanism is more natural in the Dictionary. Protégé similarly to TopBraid does not distinguish between attributes and relations. What differentiates Protégé from both TopBraid and the Dictionary is that it allows for multiple inheritances (by concepts pointing to multiple parent-concepts).

Protégé has also been created as a desktop application; however, there are works in progress focused on the web version of such application ("WebProtege") [8], which main goal is to allow for cooperative work on ontologies. The application at its current state is much less developed that its desktop counterpart or the Dictionary and there are no sings that it will be focused on sharing its resources with external applications.

5.3 PoolParty Thesaurus Server

PoolParty Threasures Server is a platform created by the Semantic Web Company GmbH and is focused on managing connected objects and is based on semantic network model. It provides governance over taxonomies, synonyms, ontologies and knowledge graphs. Knowledge stored and transformed by the platform is based on RDF, OWL and SKOS standards [9, 10].

Contrary to the Dictionary PoolParty uses SKOS to manage vocabulary; this choice allows for more flexibility in defining relations among terms but lacks in integration and ease of use for the final user. Again, more focus on standards compliance in the business layer hurts the overall usability. In contrast, current vocabulary capabilities are planned to be extended to be SKOS-compliant but without compromise in the business layer.

PoolParty Threasures Server allows for hierarchy definition; however, there are no inheritance mechanisms as in the Dictionary. It allows for definition of relations and attributes for concepts but in a different way than in the Dictionary. Attributes are defined as simple-typed values and relations are defined as triple consisting of: two classes and name (that relation can be use only between that classes and their subclass). Contrary to the Dictionary in PoolParty connection between class and concept must be create by user.

PoolParty provides an API based on SPARQL standard, which allows for connecting to the gathered data using a search engine. One of the methods provided allows for suggestions of new concepts [11]. Then, based on the sent definition a new "free concept" is created—this concept does not belong to any concept hierarchy. The advantage of this approach is having a concept being created instantly, without user mediation; however the newly created concept is not really useful until it is connected to other concepts or verified by a specialist.

6 Conclusions

Our solution is dedicated for building and developing domain vocabularies to be used for modelling and description of service systems. It is implemented as a service, and was successfully used for developing software optimization tools for transport domain. Our Dictionary has several unique features which we do not meet in other solution, it is also integrated with tools (like Task Modeler) which use dictionary for the description of optimization problems. The entire framework was already used for the development of software products dedicated for real clients (Polish transport and logistics companies).

Acknowledgements The research presented in this paper was partially supported by the Polish Ministry of Science and Higher Education and the European Union within the European Regional Development Fund, Grant No. POIG.01.03.01-02-079/12 and within European Social Fund.

References

1. Smith, B.: Blackwell Guide to the Philosophy of Computing and Information, Preprint Version of Chapter "Ontology". Blackwell, Oxford (2003)
2. http://wordnet.princeton.edu/wordnet/
3. Opengroup: http://www.opengroup.org/soa/source-book/soa/soa.htm#soa_definition
4. https://www.w3.org/2001/sw/wiki/TopBraid
5. http://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/
6. https://www.w3.org/2001/sw/wiki/Prot%C3%A9g%C3%A9
7. http://protege.stanford.edu/
8. http://protege.stanford.edu/products.php#web-protege
9. http://www.poolparty.biz
10. https://www.w3.org/2001/sw/wiki/PoolParty
11. https://grips.semantic-web.at/display/public/POOLDOKU/Method%3A+suggest+free+concept

Part III
Multi-agent and IoT Systems

Hardware Implementation of Rainbow Tables Generation for Hash Function Cryptanalysis

Jędrzej Bieniasz, Krzysztof Skowron, Mateusz Trzepiński, Mariusz Rawski, Piotr Sapiecha and Paweł Tomaszewicz

Abstract Nowadays programmable logic structures are commonly used in cryptology. FPGA implementations of cryptographic and cryptanalytic algorithms combine advantages of an ASIC and a software, offering both great data processing speed and flexibility. In this paper, we present the design and implementation of a system for rapid rainbow tables' generation. Rainbow tables are commonly used for cryptanalysis of hash functions. The presented approach shows that proposed method may compete with CPU-based approaches when performance is considered, as well as computational complexity, while maintaining low level of programmable structures' logic element utilization.

Keywords Rainbow tables · Hash function cryptanalysis · FPGA · Acceleration

1 Introduction

Programmable solutions are being increasingly used in cryptologic applications [1–3]. It is not surprising, since most algorithms used in cryptography and cryptanalysis are computationally complex, as they are based on convoluted transformations executed on vast quantity of data. Their implementations, to be efficient, require great processing power. The technological progress of FPGAs (Field Programmable Gate Arrays) observed in the last decade opens new possibilities for developers interested in cryptology. When being used for implementations of cryptographic algorithms, FPGAs offer data processing speed comparable to ASIC, while maintaining flexibility similar to the software.

One of most common cryptographic algorithms are integrity or authentication algorithms. They are used to process the confidential user data in a way preventing

J. Bieniasz · K. Skowron · M. Trzepiński · M. Rawski (✉) · P. Sapiecha · P. Tomaszewicz
Warsaw University of Technology, Warsaw, Poland
e-mail: rawski@tele.pw.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_15

unauthorized third parties from accessing it. The main assumption when it comes to creation a cryptographic systems is, that correct use of that data is only possible for the authorized users. Confirmation of this assumption determines the 'strength' or 'security' of a cryptologic method. Therefore, cryptographic architectures are based on mathematical theorems that are considered computationally complex. Such solutions require great computing power. Yet even more power is required for their cryptanalysis. This is the main reason behind using efficient microprocessor systems, specialized ASIC modules or even cloud computing in these applications.

An example of such algorithms are hash functions [4]. These are functions transforming data of any length into its fixed-size representation, called a hash. Typically, they are used in integrity methods or as a part of authentications schemes. The attacks on cryptographic hash functions relay on finding collisions of hashes. There are several approaches, such as brute-force or dictionary attacks, birthday paradox attacks, methods involving SAT-solvers and so-called rainbow tables [5, 6].

In this paper we describe an implementation of a specialized system for generation of rainbow tables for SHA-1 hash function. The proposed system consists of a hardware accelerator generating rainbow tables and software controlling the accelerator. The results of the implementation are shown, as well as its performance analysis.

2 Theoretical Background

2.1 Hash Functions

A hash function is any function that maps input digital data of any arbitrary size to an output digital data of a fixed size [7]. The output value is called a *hash* and a method to evaluate it should be computationally effective. The hash functions are widely used for easy verification of signatures. Such signatures can be used as checksums securing the data against random or intentional modifications. They can be also used as hash tables optimizing access to data storages in software applications. Cryptographic hash function must satisfy the following conditions:

- *collision resistance*—there is no computationally feasible method for generating two different messages, that have the same hash value,
- *target collision-resistance* (preimage resistance)—there is no computationally feasible method for finding a different message, that has the same hash value as the initial message,
- *one-wayness*—there is no computationally feasible method for finding the inversion of the hash function.

Hash functions' security is based on an assumed resistance to well-known cryptanalytic attacks. However, there is no mathematical proof that the cryptographically secure hash functions exist at all. Many weaknesses of hash functions historically regarded as secure were found, e.g. MD2/5 or SHA1.

Another important application of hash functions is data integrity. Comparison of hashes of two different files shows, if they were subjected to any changes. Many digital signature algorithms operate on messages' hashes, so authenticity of any of them could be verified anytime. Hash functions are also used for password verification. In databases, users' passwords are stored as hashes in order to increase data security. Process of user verification is simple and requires comparing two hashes—one derived from user password input with one stored in a database. Finally, an example of hash functions usage are cryptocurrencies. In 2009 the Bitcoin was introduced by a person (or a group of people) named Satoshi Nakamoto. The generation of bitcoins is commonly called mining as an analogy to gold mining. It practically equals to computing SHA-256 hashes [8].

2.2 Rainbow Tables

Rainbow tables were proposed in 1980 by Hellman [9] as a method for cracking symmetrical ciphers. In this paper we propose applying Hellman's idea to the cryptanalysis of hash functions—to find their inversions. The aim of the method is to reduce the memory requirements of the algorithm at the expense of computing time or vice versa—this is called *time-memory trade-off* [10]. The idea is to store only some password-hash pairs, whereas all missing information could be computed on the fly.

Let H be a hash function and let d be an input data. $H(d)$ means the hash of the input data. Let us consider easily computable reduction function R, that for any hash $H(d)$ designates the $t = R\ (H\ (d))$ (with length of the analyzed data). A reduction function may be, for example, a function that converts binary representation of a hash into ASCII characters.

We could create composition of these functions: $F(d) = R(H(d))$. Then a finite sequence of successive iterations of the function $F(d, F(d), F(F(d)), \ldots, F^{(n)}(d))$ could be build (where $F^{(n)}(d)$ is a n-fold function composition of F). Only the initial password of such chain and the last item: $F^{(n)}(d)$ are stored. The other useful information is the length of the chain. Next, a dictionary of such pairs is created: $T = (d_i, F^{(n)}\ (d_i))$, where $i = 1, 2, \ldots, m$, for various values of the starting d_i. This collection is commonly known as *rainbow tables*. The following listing shows the search algorithm for a password d with given hash $h = H(d)$ using the rainbow table:

```
input:  T - rainbow tables, h - hash, H - hash function,
R - reduction function;
output: password d that H(d) = h;

search for d := R(h) in a set Z := {F^(n)(d_i),i = 1,..., m}
while (d is not inside Z) do
  d := F(d);
  find d in set Z;
if (d is inside Z for d_k) then
  compute value of password in this chain starting from d_k
  that H(d)=h;
if (d found) then
  return d
else
  return false;
```

This algorithm helps in a passwords' recovery process, but does not guarantee the success. An appropriate selection of parameters m and n has an great impact on the success rate. If system has sufficient memory, parameter m could have a higher value and if system has enough computation power then parameter n could have a higher value.

3 The Concept of a System for Rainbow Table Generation

The concept of the system for hash table generation has been schematically presented in Fig. 1. The system is composed of two main parts: hardware accelerator implemented in FPGA of a platform in form of extension card mounted in

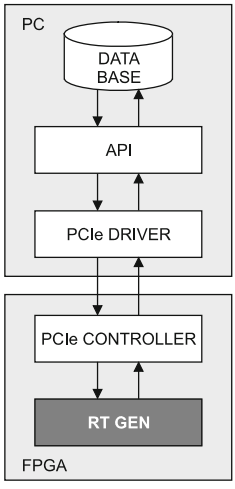

Fig. 1 Block diagram of the system for rainbow table generation

PCIe slot of a PC and software application executed on this PC, which is the interface and control of the accelerator.

As a hardware platform a DE5-Net solution from Terasic has been used [11]. It is a PCIe board equipped with Stratix V GX FPGA 5SGXEA7N2F45C2 chip from Altera. The board has been installed in PC, on which an application written in C++ has been executed, that implements the control part of the system.

The main task of software part is implementation of PCIe interface for sending an initial passwords d to hardware accelerator and receiving a hash $F^{(n)}(d)$, that is the last element of a chain.

Hardware accelerator is composed of PCIe controller module, that implements a DMA transfer between the accelerator and the PC memory, and the *RT GEN* module, that is the core element of the accelerator for the generation of $F^{(n)}(d)$ elements of the rainbow table chains.

There exists a lot of rainbow tables, which differ in the password length, the set of valid password characters and the length of the chains. The proposed architecture allows configuring the RT GEN module in such a way, that a given type of rainbow table can be generated. In this paper such architecture has been implemented, that generates rainbow tables for passwords of the length of up to 12 characters from the set containing a–z, A–Z, 0–9 and !@#$%^&* special characters—totally 70 possible characters.

Figure 2 shows the scheme of the accelerator processing unit *RT GEN*. Passwords, sent from software layer, are stored in input queue *FIFO_IN*. Then these passwords are processed by *SHA1* module, that generates a hash, that is the beginning of the chain. These hashes are stored in *FIFO_SHA1* queue. They form input data of *RED_SHA1* module, that iteratively generates subsequent elements of the chain. Hashes $F^{(n)}(d)$, that are the final elements of chains for given password d are stored in output queue *FIFO_OUT*. This queue is read by software part of the system.

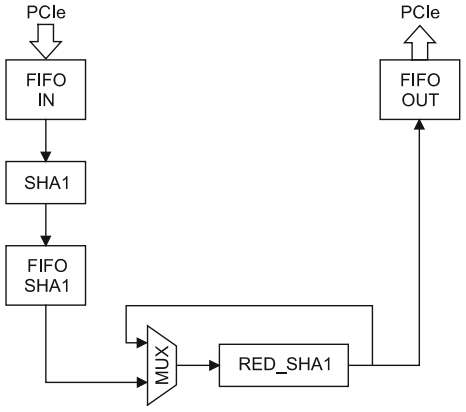

Fig. 2 The block diagram of hardware accelerator *RT GEN*

4 Implementation

The core element of the proposed accelerator is a hardware unit computing hashes for a given input. It is used in *SHA1* and *RED_SHA1* modules. A hardware implementation of hash function, such as SHA-1, may be challenging in comparison to software implementation. Hash functions are in general iterative algorithms, which are designed for sequential CPUs—general-purpose processors are usually clocked with significantly higher frequencies than FPGA's clocks. However, FPGAs are capable of processing multiple data at the same time.

4.1 SHA1 Module

The main element of SHA-1 function is a transformation called *round*. Round consists of several bit-wise and word-wise operations, such as logic functions and modulo addition. Processing one 512 bit word into 160 bit hash takes 80 iterations of such transformation [12, 13].

Figure 3 shows hash function block diagram designed for this accelerator. 80 iterations of SHA-1 algorithm were divided into 5 blocks of 16 iterations. Blocks were connected using pipelined registers, that allow processing of 5 passwords simultaneously. This architecture generates a new hash every 16 clock cycles.

Linear feedback shift registers are used for generation of words used in every round. These words are then processed into a hash during 80 iterations of SHA-1 algorithm. Due to processing 5 words simultaneously, 5 LFSRs are required. Each of them produces 80 words in one work-cycle, while each round processes 16 different words. Words must be routed to respective round blocks accordingly. It was achieved with a switching fabric, which cyclically alters the connections between registers and round blocks.

SHA1 module produces a new hash every 16 clock cycles. In order to obtain a hash every clock cycle, *SHA1_PARALLEL* block has been implemented (Fig. 4).

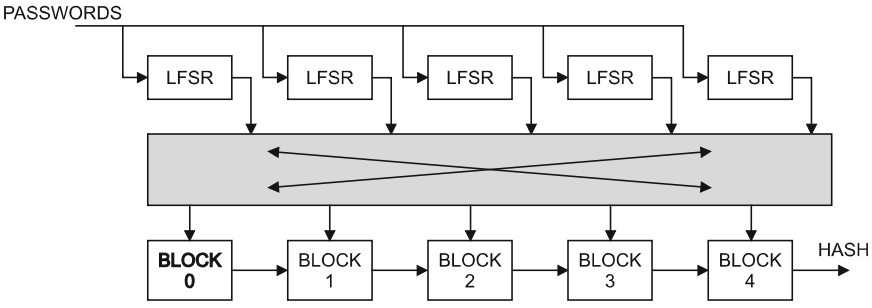

Fig. 3 Schematic representation of the module implementing SHA-1 function

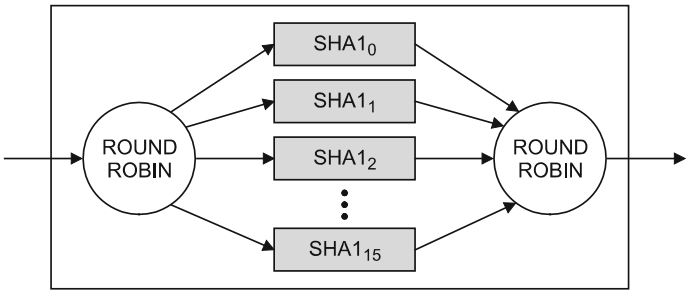

Fig. 4 Parallel hash generation block

It utilises 16 *SHA1* blocks in parallel, as well as two round robin schedulers. Every clock cycle a new password is passed into subsequent *SHA1* block, while a new hash is received. This module processes 80 hashes simultaneously—5 hashes in each of 16 *SHA1* blocks.

4.2 Chain Generator

The main element of *RED_SHA1* chain generator module is a reducing-hashing block performing $F(d) = H(R(d))$ operation. Figure 5 shows the block diagram of this chain generation module.

Each chain element contains one reduction block, that reduces the input hash into valid password, and one *SHA1_PARALLEL* block returning a hash of password generated by reduction module. The reduction consists of a function that selects 128 bit groups from 160 bit input hash. Every 8 bit word is mapped into an ASCII character. As the set of possible 8 bit words is larger than the set of acceptable characters, reductions was implemented as a truth table, that returns the same ASCII character for several different words. This allows modelling the probability density function of character occurrence and in this way fit system's output to required conditions. Reduction function has been implemented as a combinational block with low resource utilization.

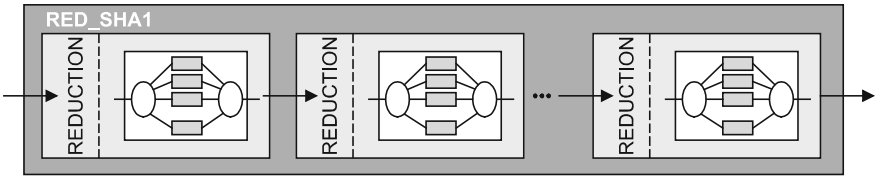

Fig. 5 Block diagram of chain generator

As a result of application of pipelining and parallelism, *RED_SHA1* block can process 80 passwords simultaneously, thus it produces one hash per clock cycle. Such module represents one stage of rainbow table generation chain. The proposed architecture is parameterized, therefore chain length can be modified accordingly to logic resources available in selected FPGA architecture. Extending the length of the chain results in higher logic utilization, but it reduces processing time. For example, with only one stage, it takes 10,000 iterations to complete the chain of length of 10,000. Pipelined composition of 5 such stages reduces iteration count to 2000 and increases system's throughput.

4.3 In/Out Interface

The communication between PC host and FPGA platform is implemented using PCIe interface. In the presented system the *TERASIC PCIe IP* module has been used, that is designed by DE5-Net platform developer—Terasic. This module uses PCI Express IP Core generated with Altera CAD tool Thanks to transceivers embedded in the Stratix V GX FPGA device, the whole PCIe interface module has been implemented in programmable device. This module implements memory-mapped read-write operations, as well as DMA (*Direct Memory Access*) mechanisms that allow transferring data between hardware accelerator and PC host with the throughput of 2 Gbit/s. From the software side the solution delivered by Terasic used Jungo driver and through the system libraries gives the access to a simple API for implementing communication in the software application. In Fig. 6 a schematic diagram of FPGA platform/PC host communication mechanisms is presented.

An interface enabling communication of accelerator with the *TERASIC PCIe IP* was implemented using two FIFO queues—on the ingress and the egress of the

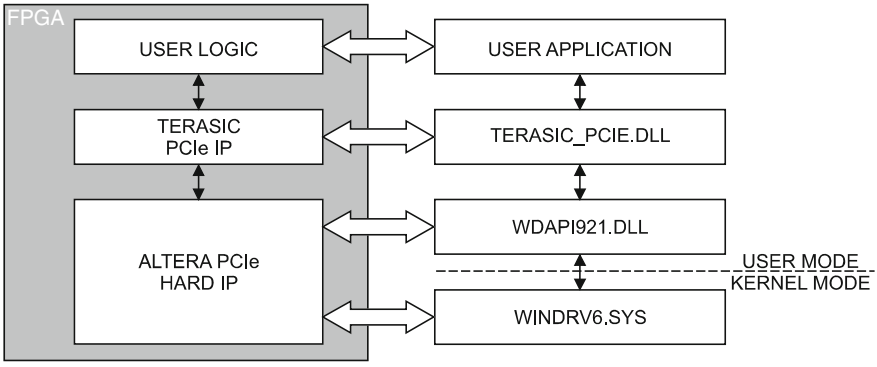

Fig. 6 Schematic diagram of DE5-Net/PC communication mechanisms

module (Fig. 2). Read and write operations are controlled by different clock signals, in order to synchronize the PCIe interface of the system and core data processing unit, that operate in different clock domains. The ingress FIFO stores hashes calculated by SHA1 from passwords received from the input port of the accelerator. The egress FIFO stores generated hashes, that are sent to the managing application. Both queues are controlled by the FSM (*Finite State Machine*).

5 Results and Testing

All components of the system were modelled in Verliog HDL and implemented in Stratix V GX 5SGXEA7N2F45C2 FPGA using Altera Quartus 15.0 CAD software. Table 1 depicts the results of the implementation in terms of FPGA's logic resources utilisation used for an instance of the accelerator.

Single SHA1 function uses less than 1700 logic elements, so it was possible to use 16 such modules to realize parallel version of SHA1_PARALLEL block.

The *RED_SHA1* block has been implemented as a chain consisting of stages containing reduction function and *SHA_PARALLEL* module. The designed architecture allows generating a chains with lengths depending on logic resource available in the target FPGA system. The resources of Stratix V GX 5SGXEA-7N2F45C2 allowed for implementing the chain of 5 stages, what considerably increased the system's efficiency. Table 2 presents the parameters that describe the efficiency of the proposed solution.

The timing analysis was performed with use of QuartusII CAD software. The evaluated *RT GEN* accelerator's maximum frequency was 162 MHz. It means that it can generate single hashes at the rate of 162 Mhash/s, what gives throughput of 26 Gbits/s. The entire accelerator processes 81 thousand passwords per second—which results in throughput of 13 Mbits/s for password of length of 12 characters.

System equipped with a single hardware core is able to generate rainbow tables for a 100 GB password database in about 17 h. It is also worth noticing, that

Table 1 Results of the implementation

Module name	Number of LE
TOP	140,317
– PCIe controller	21,670
– RT GEN	118,647
– RED_SHA1	28,168
– SHA1	1699

Table 2 Parameters describing the *RT GEN* accelerator efficiency

f_{max}	162 MHz
SHA1	162 Mhash/s (26 Gbit/s)
RT_GEN	81 Mhash/s (12.96 Mbit/s)

equipping more units would linearly increase the throughput of the system and decrease the rainbow tables' generation time. As an example, 10 units would generate rainbow tables for a 100 GB password database in less than 2 h.

Validation of implemented system was performed on the basis of functional simulation using Mentor Graphics ModelSim software. Every component and the system, as well as the accelerator as a whole were verified through simulations using test vectors generated by the software reference model.

Figure 7 shows functional simulation of *SHA1* module computing a hash for a password given on the its input. In the figure the procedure of calculating a hash function is presented for the following password: **08HScjpw08HS.** After 80 clock cycles the module returns the correct hash: **ac253798a326a09786cea1d0542df53ecacc81a5.** At beginning of each cycle, that takes 16 clock cycles, the result of next processed password is returned.

It was possible to receive new hash every clock cycle, thanks to using 16 *SHA1* functions in parallel in one computing block (Fig. 8). On the input there are 2 subsequent passwords: **iiifXMA4zsmf** and **08HScjpw08HS**. First of them was inserted in 22nd cycle (counter mod 16: 6, mod 5: 1). After 80 clock cycles correct results appeared on the output: **0067c5ca92334795da35ef9e479bcc7bc047d744** and **ac253798a326a09786cea1d0542df53ecacc81a5**.

Passwords in every stage of the chain (Fig. 5) are calculated by reduction function performed on given hash. Figure 9 shows how the reduction works. Block calculating reduction function is combinational, thus it returns a result in the same clock cycle. As can be seen, from 160 bits of hash there are chosen twelve 8-bit words, that are casted on set of 70 characters in ASCII encoding.

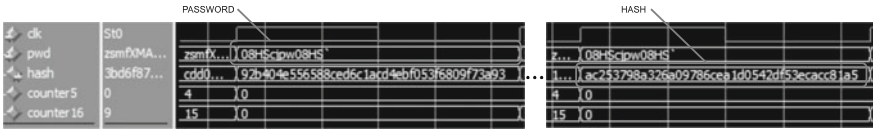

Fig. 7 SHA1 block simulation

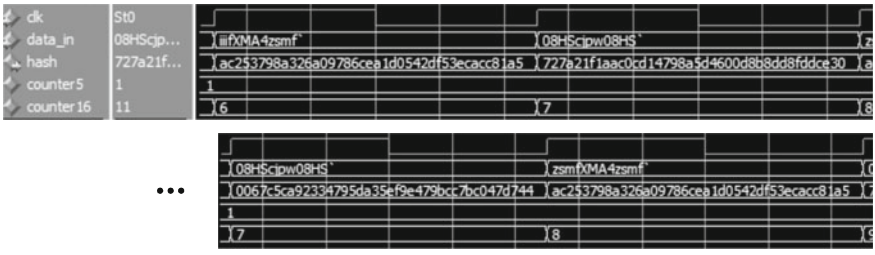

Fig. 8 SHA1_PARALLEL simulation

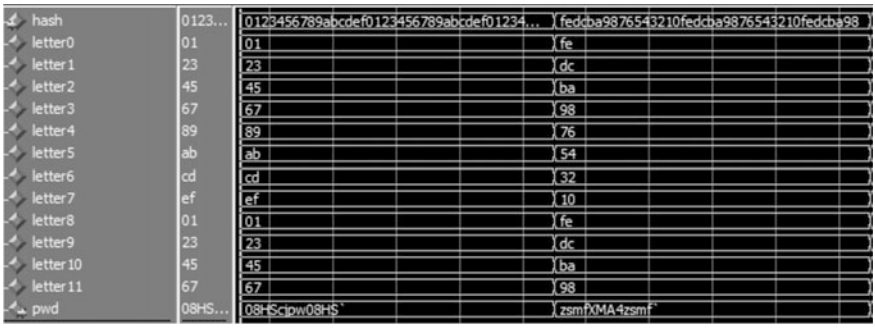

Fig. 9 Reduction block simulation

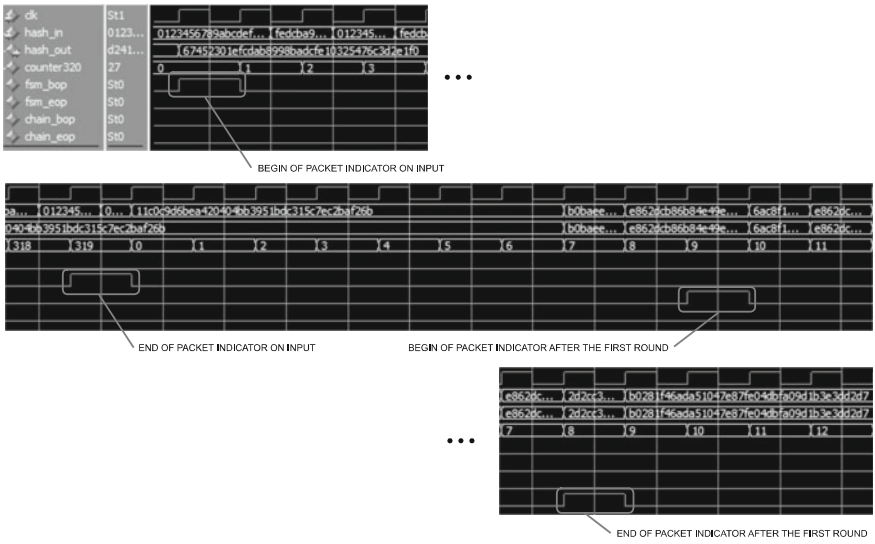

Fig. 10 Chain simulation

Figure 10 shows functional simulation of chain module generating rainbow tables. For each burst of hashes there are shown start and end markers. Their length depends on the size of the chain (here 320 hashes). Markers are generated by FSM, when hashes are being received from FIFO. Then hashes are looped several thousand times in the chain and passed to egress FIFO afterwards. On the waveform it is shown that chain's output data are passed to its input. Markers are showing up again at the beginning and on the end of the chain.

6 Conclusions

In this paper the FPGA-based accelerator for generating rainbow tables has been proposed. The implemented solution processes 81 million passwords per second. That gives a throughput of ∼13 Mbit/s, which allows generating the rainbow tables for a 100 GB password database in about 17 h, using only one accelerator unit. Our solution is highly scalable and using more units results in linear increase of processing performance. What is very important, it can be easily configured in terms of chain length and even hash function used. The developed solution allows building a hardware/software system that generates rainbow tables for password databases.

References

1. Koç, Ç.K. (ed.): Cryptographic Engineering. Springer, Berlin (2008)
2. Rodríguez-Henríquez, F., Saqib, N.A., Díaz-Pérez, A., Koç, Ç.K.: Cryptographic Algorithms on Reconfigurable Hardware. Springer, Berlin (2007)
3. Hulton, D., Pellerin, D.: Accelerating cryptography with FPGA clusters. http://mil-embedded.com/articles/accelerating-cryptography-fpga-clusters/
4. Shi, Z., Ma, C., Cote, J., Wang, B.: Hardware implementation of hash functions. In: Tehranipoor, M., Wang, C. (eds.) Introduction to Hardware Security and Trust. Springer, Berlin (2012)
5. Stevens, M.: Attacks on Hash Functions and Applications. CWI, Amsterdam (2012)
6. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: "Time-memory trade-off Attack on FPGA", platforms: UNIX password cracking. In: Bertels, K., Cardoso, J.M.P., Vassiliadis, S. (eds.) ARC 2006, LNCS 3985, pp. 323–334. Springer, Berlin (2006)
7. Pieprzyk, J., Hardjono, T., Seberry, J.: Fundamentals of Computer Security. Springer, Berlin (2003)
8. Oliveira, S., Soares, F., Flach, G., Johann, M., Reis, R.: Building a bitcoin miner on an FPGA. In: XXVII SIM—South Symposium on Microelectronics (2012)
9. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theor. **26**(4) (1980)
10. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Advance in Cryptology—CRYPTO 2003, LNCS 2729, p. 617 (2003)
11. Terasic: DE5-Net FPGA Development Kit. http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=158&No=526
12. Pieprzyk, J., Sadeghiyan, B.: Design of Hashing Algorithms. Springer, Berlin (1993)
13. FIPS 180-4: Secure Hash Standard (SHS). http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

An Implementation of an Ad Hoc Mobile Multi-agent System for a Safety Information

Yasushi Kambayashi, Takushi Nishiyama, Tomofumi Matsuzawa and Munehiro Takimoto

Abstract Mobile Ad hoc Network (MANET) is a network that connects mobile devices without any central control. Since MANET is an infrastructure-less network that consists of only mobile devices, it should be convenient means to construct temporary networks as a part of contingency plan (when a disaster occurs). In this paper, we report our experience that we have obtained through our design and implementation of a mobile software agent system running on mobile ad hoc network. In order to demonstrate the usefulness of the mobile software agent system, we have implemented a safety information collection system for sufferers of a disaster as an example application. Our experience suggests that the combination of mobile software agents and ad hoc network open the new horizon of temporary communication system without infrastructure.

Keywords Ad hoc mobile network · Delay-tolerant network · Multi-agent system · Mobile software agent · Disaster information system

Y. Kambayashi (✉) · T. Nishiyama
Department of Computer and Information Engineering, Nippon Institute of Technology, 4-1 Gakuendai, Miyashiro-Machi, Minamisaitama-Gun, Saitama 345-8501, Japan
e-mail: yasushi@nit.ac.jp

T. Nishiyama
e-mail: c1095350@cstu.nit.ac.jp

T. Matsuzawa · M. Takimoto
Department of Information Sciences, Tokyo University of Science, 2641 Yamazaki, Noda, Chiba 278-8510, Japan
e-mail: t-matsu@cs.is.noda.tus.ac.jp

M. Takimoto
e-mail: mune@cs.is.noda.tus.ac.jp

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_16

1 Introduction

Today, many people are carrying multi-functional and high performance mobile devices. Those mobile devices, typically smartphones, came to be equipped with such functions as GPS receiver, gyro sensor, and a camera as well as the mobile communication system such as wireless LAN and Bluetooth as a set of standards. In fact, mobile devices are tiny computer with many applications that utilize those functions. Despite the proliferation, miniaturization, and functional improvement of mobile phones, however those features may become useless in a massive disaster. In a recent disaster, mobile communications became extremely difficult in a wide area due to the destruction of base stations, power failure, and communication congestion. Therefore information gathering and route search by mobile devices were also disabled. Under such circumstances, it must be helpful if an impromptu communication system with the surrounding independent mobile devices, such as smart phones, is enabled to share safety information. If ad hoc communication becomes possible by a relay of many smartphones, the safety information can be conveyed to remote people through information exchange solely by smartphones.

We have conducted several research projects using mobile software agents that include collaborative movements of robots in multi-robot systems and effective searches for resources in overlay networks [1–4]. The effectiveness of mobile agents in ad hoc networks and usefulness of mobile agents in constructing ad hoc networks are theoretically recognized [5]. A common point among those studies is to demonstrate that communication can be performed by using mobile agents in a decentralized environment. We have designed and implemented a simulator of the Return Route Support System in by applying our previous studies [6]. The system searches and presents return routes in the event of a disaster using mobile agents on a network consisting of only smartphones. We have confirmed that mobile software agents that autonomously determine the moving behavior are effective in unstable communication environment such as the after-disaster situation. However, there is a limitation in the experiment on a simulator. Stable communication is not always possible in a highly fluid environment. In addition, since the number of simultaneous connections is strictly limited in the present communication systems with mobile devices, it is impossible to construct a network that enables constant communication. Repeated connection and disconnection between mobile devices are required to convey mobile agents. Further, the actual duration time required to send data to the destination must be measured only by an actual working system on real machines.

This paper reports our experience that we have obtained through an implementation of an agent system based on the ad hoc communication. This system operates on the Android OS and can communicate with surrounding tablet computers using the wireless LAN to construct a temporary network. The user of this system can produce a mobile agent with short message and dispatch it to other devices without any communication infrastructure. We have designed and implemented the ad hoc mobile agent system, and have conducted experiments on an

application that exchanges safety information using this system. The mobile agent searches for the safety information server on a Windows PC that is working within the network and moves to it. We have constructed the system so that mobile agents are migrating to the PC that serves as a safety information database and a bulletin board system.

The structure of the balance of this paper is as follows. The second section describes the background and the objective of our project. The third section describes the design of the ad hoc mobile software agent system and explains the imaginary application and a scenario of use. The fourth section describes the implementation of the ad hoc mobile software agent system on the real machine. The fifth section demonstrates the effectiveness of our system through experiments. The section also discusses the limitation of the current system and denotes the future directions. Finally the sixth section concludes our discussion.

2 Backgrounds

There exist several services that provide safety information in the event of a disaster. A famous service is the safety information program broadcasted by NHK on television and radio in the event of a large-scale disaster. This service receives safety information at the call center and broadcasts the information. NHK makes the information be available on the Internet web site too [7]. There is also a safety information system called Safetylink24 provided by Askul Inc. [8]. Those services are premised on the Internet connection in principle and cannot be used under a condition where communication with the base station is disabled. It must be helpful if a small amount of information such as the information about the safety of friends and families can be known through another means. In such a case, a means of constructing a temporary network have to be provided only by mobile devices without any infrastructure.

2.1 Mobile Ad Hoc Network and DTN

A network that connects communication devices using only the wireless communication mechanism mounted on mobile devices and not using any central control is called Mobile Ad hoc Network (MANET). This communication method came to be actively investigated since the middle of 1990s when mobile phones came to proliferate. In the late 2000s, more flexible and powerful applications came to be developed due to the advent of smartphones. As a result, a much more flexible communication system than MANET appeared as the delay-tolerant networks (DTN) [9, 10]. DTN is a method of transmitting data to the target node by storing and forwarding the data among nodes. There also exists a study on changing the communication mode depending on the situation with MANET and DTN combined [9].

2.2 Mobile Software Agent

We have conducted several projects using mobile software agents. A mobile agent is a program that processes data while migrating among computers. The target destination can be autonomously determined by the mobile agent. Examples of the mobile agent application include the search for resources using a multi-robot system. The use of the mobile agents has achieved an effective search for resources by sharing the information obtained by a robot and instructing and controlling each other [1, 2, 11]. As Comer stated in his book, few technology has been established as of 2014 that continuously maintain network connection among many moving mobile devices [12]. In constructing a network with mobile devices, the maximum number of Android-based mobile devices that can simultaneously connect is three by using Wi-Fi Direct and seven by using Bluetooth. Therefore the real system may not work on real machines as on the simulator. We need to construct a real system by using only mobile devises and communication system equipped on the machines.

3 The Mobile Agent System

The imaginary application that we have in our mind is a safety information system that transmits short messages with location information (GPS position) in a mobile agent (hereinafter we refer this as the safety information agent) from smartphones of sufferers and collect the information in the safety information server. The safety information system consists of many mobile agents and one server. Each user dispatches mobile agent from his or her mobile phone that conveys appropriate safety information, i.e. s short “hello” message and GPS information. The mobile agent hops to other mobile devices, e.g. smartphones, by using ad hoc network connections and tries to reach to the server. The safety information is accumulated on the server for further use. In order to move to the destination, the mobile agent senses the circumstance and chooses the most promising mobile device to hop. Since the current Wi-Fi direct can connect to only three other smartphones simultaneously, the mobile agent establishes the connection just before transferring itself and disconnects immediately after transferring.

The reason for using a mobile agent is that mobile agents can bring the current state of the device (battery, current location, etc.) and the damage situation that is constantly changing in the event of a disaster, so that they can propagate the desired information as well as can determine their behaviors autonomously.

3.1 Connection Mode

Given the limited number of simultaneous connections of the communication API of Android, we have selected an ad hoc communication method that repeats connection and disconnection for transferring mobile agents. In other words, communication will be performed by storing and forwarding data similarly as in DTN.

One of the advantages of the mobile agent is that it can determine the destination when it performs transfer. In the conventional routing protocol, when communication request occurs, the protocol software confirms the access points and establishes the connection. In the MANET environment, however, communication must be performed by repeating connection and disconnection as the mobile agent moves toward the destination, because the location of the target computer is unknown and the number of connections is limited. In our agent system, the mobile agents are designed to decide the next transfer destination using the positional information. In other words, the mobile agent controls the route not in the Internet layer as the conventional routing protocol but in the application layer. For this reason, the mobile agent will move from one mobile device to another in the following processes. Each participating mobile device has one agent server as a house keeper.

1. A mobile agent requests migration to the agent server.
2. Since the agent server is sensing the circumstance, it gives the agent a list of candidate next-hops.
3. The mobile agent chooses one next-hop, and notifies the target device to the agent server.
4. The agent server establishes a connection to the selected mobile device, and sends the mobile agent to the agent server on the target device.
5. After confirming the arrival of the agents, the agent server disconnects the connection.
6. Go back to step 1, as necessary.

If several mobile agents intend to move to the same device or different devices, the above processes are repeated.

3.2 Reduction of Communication Traffics

The repetition of the above processes makes the connection time and frequency increase. This may cause problems in unstable communication networks such as MANET and DTN. The mobile agent may not be able to move due to network congestion and battery consumption. In order to mitigate this problem, we take advantages of the idea of the hierarchical agent system proposed by Satoh [13]. In the current system, the mobile agents move to the same device can be bundled in one batch and are transferred at once. More specifically, a container mobile agent is generated that contains several mobile agents as shown in Fig. 1. This method

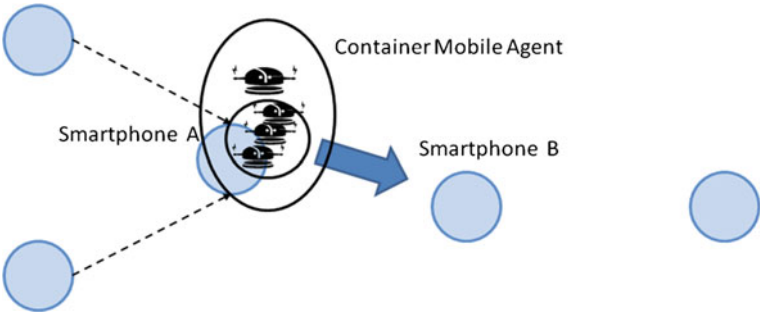

Fig. 1 Container agents bundles multiple mobile agents and conveys at one

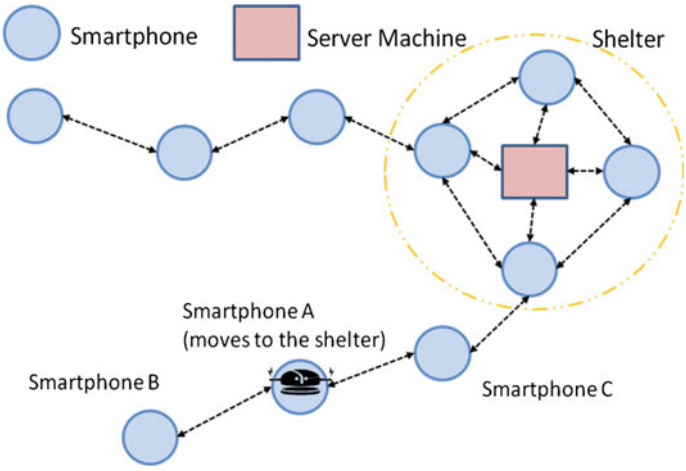

Fig. 2 An imaginary scenario

enables multiple mobile agents to be transferred at once. Once the transfer completes, the container mobile agent will be discharged and the mobile agents included in it will be unbundled into independent mobile agents.

The imaginary scenario and environment is the course of evacuation after occurrence of disaster. In such a situation, many people are heading for a specific evacuation shelter as shown in Fig. 2. Therefore this method should be effective. The mobile devices in the scene are connected as shown by the arrows in Fig. 2. In and near evacuation shelter, people can enjoy stable communication with each other by the MANET, and, a stable long-distance communication line may be established for emergency use. The people in the course of evacuation, however, cannot easily connect to others due to the small number of people in their surroundings. Therefore, the assumed topology may be in a tree with the evacuation shelter as a root.

We assume this environment as a precondition. The safety information agent needs to move to the server installed in the evacuation shelter. Suppose that the safety information agent is transmitted by the smartphone A in the course of evacuation. While sensing the radio signal in the circumstance, the smartphone A finds smartphones B and C. The smartphone A, however, cannot determine to which smartphone it should transfer the safety information agent. One possible method is to retain the initial positional information and the name of accessed devices, extract not-visited devices, and transfer the safety information agent to devices among them at random. If it goes well, the safety information agent can move to the smartphone C and, in turn, to the server machine in the evacuation shelter. However, if the safety information agent is transferred to the smartphone B in the opposite direction, it will come to a dead end; it has no choice but returns to the smartphone A, and then transfers itself to the smartphone C. Other methods may include transferring of the safety information agent away from the initial position or transferring several copies of safety information agent. However, the latter method may create unproductive communication traffics.

In order to solve this problem, the proposed agent system adds the number of sensed terminals in the circumstance in the SSIDs of the smartphones by taking advantage of the fact that there are many devices near the evacuation shelter and the number of mobile devices decreases with the increase of the distance from the evacuation shelter.

The reason why we set that simple method to direct the mobile agents is that the safety information agents are supposed to move to nearby shelter. As the next stage of our development, we are designing a new safety information system where the mobile agents are supposed to move to family members' computer instead of the server machine at the nearby shelter. Then the mobile agents would take advantage of GPS position to determine the next hop.

4 Implementation

In order to implement the proposed system, we have employed Nexus 7 Android tablets developed by Google. The operating system is Android OS 4.3 with IEEE 802.11 b/g/n as Wi-Fi. Even though the Android OS supports Bluetooth too, we tentatively decided to use IEEE 802.11.

4.1 System Configuration

The agent server as well as mobile agents is implemented in Java. While the virtual machine that runs on the Android OS is different from the formal virtual Java machine, the system is separated from the OS area using API. Thus, it is easy for us to transplant the system in a Windows machine as a server. Both Mobile Agent and

Container Mobile Agent are subclasses of the abstract class Agent. When they are instantiated, the Agent Server gives one thread for each instance and makes them run. The descriptions of the components in the agent system are as follows:

Agent Server is the house keeping program that discharges the roles such as controlling, scheduling, and input/output of agents, and producing container mobile agents. The agent server assigns an execution thread to each mobile agent.

Mobile Agent is the program that serves as the mobile agent. It is designed to carry a short message. Even though the agent is designed to be mobile, it can be served as a static agent too, as necessary.

Container Mobile Agent is produced by the agent server when there are two or more mobile agents with the same destination in the waiting queue. When the transfer completes, the container mobile agent discharges the mobile agents and disappears. The container mobile agent is used only for transfer and has no other processing functions. In MANET, the increase of the connections between mobile devices leads to performance degradation. By using the container mobile agent, it is possible to reduce the number of connections to the same destination.

4.2 Implementation of the Agent

The implementation of the mobile agents is enabled by inheriting the Agent class that forms the basis for the implementation. The Agent class includes several types of callback functions as interfaces as well as Serializable interface. Those functions relate to such events as transfer of a mobile agent and success of connection with a specific device. Some of the functions are listed in Table 1.

Table 1 Examples of functions

Function name	Description
run()	The main function; inherited from the Java *Thread* class
arrival()	Invoked when the mobile agent reaches to a device
leave()	Invoked when a mobile agent departs from a device
suspend()	Invoked immediately before shifting to suspension
changedPeerState()	Inquires whether the surrounding condition changes
receivedAnotherAgent()	Invoked when another agent accesses the same device
discoverPeers()	Requests searching the candidate of next-hops
getPeers()	Obtains the candidate of next-hops
getDestination(d)	Selects the destination
connectTo(p,o)	Request connection to the destination specified with IP address
succeedConnection()	Invoked when the connection with a specific device is established
failedConnection()	Invoked when connection with a specific device fails
succeedSession()	Invoked when data transmission succeeds
failedSession()	Invoked when transmission fails in such a case as move of the user

```
public void run()
  LinkedList<AgentPeerInfo> peers;
  //Acquire a list of devices in the surrounding
  while((peers=getPeers()) == null) {
  //If there is no peer, run searching
    discoverPeers();
  }
  //Determine and acquire a transfer destination
  AgentPeerInfo p = getDestination(d);
  //Connect to the device intended to move
    connectTo(p, this);
}
public void succeedConnection() {
  //Move to the access point
  move();
}
```

Fig. 3 Example of mobile agent program

The mobile agents, which can be bundled in a container agent as discussed in Sect. 3.2., need to inherit the bundle interface. Inheriting this interface enables optional description of the isBundled() function, which is invoked immediately before the mobile agents are bundled, and the isReleased() function, which is invoked when the mobile agents are released from the container agent. An example of a code for a mobile agent, which senses smartphones in the circumstance and make itself transfer to one of them, is depicted in Fig. 3.

There are two types of migration for mobile agents. One is strong migration and the other is weak migration. In the strong migration, a mobile agent carries the environments, i.e. local and stack variables as well as program counter so that the migrated agent can resume its execution precisely the same execution point before migration. In the weak migration, a mobile agent has a particular starting point so that all the migrated agents start that same point. Even though we have implemented several mobile agent systems with strong migration, we decided to employ the weak migration in this system, because the mobile agent has a simple role; carries short message. In this implementation, we define a class with variables such as “name”, “contact” and “address”; the mobile agent system carries this serialized instance.

5 Experiments and Discussions

Our implementation of the ad hoc mobile software agent proves that it is feasible to construct mobile agent system just on the ad hoc network. The efficiency, however, must be considered as another important issue. In order to discuss the efficiency of our mobile agent system, we have conducted a feasibility study of our container mobile agent mechanism.

First, we measured the duration time one user sends an agent to the server in one hop, and then we measured the duration time each of the ten users sends one agent to the server in one hop.

Second, we experimented of the case that ten users participated and stationed at various locations. We measured the duration time that an agent was sent by a remote user and it reached to the server by migrating other users' mobile devices, and the duration time that each of the ten users sends one agent and all the agents reached to the server. Each mobile agent retains a list of mobile devices it visited to prevent moving to the same devices it has already visited so that it can avoid to pursue a cycle. When the agent server on a mobile phone finds there are more than two mobile agents on the phone, it generates the container mobile agent to bundle them to earn efficiency. The results are as shown in Table 2. The time indicates the mean value of ten trials, respectively.

In the experiment, we assumed that there was one server on a desktop computer to store information in the evacuation shelter as the destination of mobile agents. We are aware of the limitation of our experiments. We have employed only ten tablet computers so that the scale is not practical enough. Still we believe we have achieved to demonstrate that it is at least feasible to construct a mobile agent system consists only mobile phones connected with only ad hoc network.

In order to construct a practical system, we have to overcome several challenges. The first challenge is that the traffic of mobile agents heading to the evacuation shelter may increases when the number of people in the course of evacuation increases. This results in concentration of communication traffic and may cause excessive load on mobile phones in the course of evacuation. The current system does not pay much attention of the congestion and power consumption other than the container mobile agent. Many routing methods are proposed and discussed to preserve energy consumption. We need to develop more sophisticated routing methods with the consideration of congestion and power consumption in order to construct an actual system. In addition, this experiment was performed only with

Table 2 Results of the experiments

	Number of users	Time (s)
Time required for one hop	1	1.6
Time required for one hop	10	2.5
Time for an agent reaches to the shelter	1	8.1
Time for all the agents reach to the shelter	10	31

the safety information agent and the container mobile agent that share the same purpose. Our system does not only aim to safety information system but also aim to general purpose mobile agent system. In implementing other applications, a more elaborate routing method needs to be developed in order to prepare and transfer a variety of mobile agents.

The second challenge is that while the mobile agents migrate, they make use of the GPS information and the number of mobile devices they sense in the environment. If a group of evacuation people moves to the direction opposite to the evacuation shelter, the safety information agent may follow that group of people's devices. Mobile agents need to trace back to reach the true destination. Possible solutions for this challenge include a routing method that navigates the safety information agents to the correct direction. This can be achieved by the server emitting some kind of beacon to attract safety information agent. We have implemented pheromone as another kind of mobile agent for the purpose of the beacon [4, 6]. Another solution may be to prepare several safety information agents for single safety information and to transmit them in different directions. However, this method spreads more than necessary traffics of mobile agents and may not be very useful under the disaster condition with limited resources.

The third challenge is that the server installed in the evacuation shelter used in the current system is a simple one only for collecting and storing safety information. In order to use the collected information effectively, we needs to consider some linkage with other systems that post the information on twitter or SNS. The experiments we have performed have the purpose of sharing safety information in the event of disaster and thus did not pay much attention of the security of the mobile agent and personal information. This may be also a challenge.

As the second stage, we are designing a new mobile agent system where mobile agents try to move not only to the evacuation shelter but also to the home PC of the users' family members. In that case, the mobile agents should make use of GPS information to move. In the system, each mobile device not only transmits its SSID and the number of devices it senses but also transmits the geographical position so that a mobile agent can move to mobile devices that geographically appropriate direction to the home.

6 Conclusions

We have proposed an ad hoc mobile agent system and implemented it on a set of mobile devices. The mobile devices operate with the Android OS and the mobile agent system directly communicates with other mobile devices to transfer mobile agents using Wi-Fi Direct. As an example application, we construct a safety information system where the mobile agents convey short messages in the case of disasters. We believe that the flexibility of mobile software agents is suitable for ad hoc network. Kawaguchi et al. have demonstrated the usefulness of mobile agents by constructing mobile ad hoc network systems through implementing DSR

protocol [14, 15]. We plan to implement dynamic AODV-like protocol for the same purpose [16]. In constructing a practical ad hoc mobile agent system, there are challenges in such perspectives as the maintenance of the network, routing, and power consumption. The mobile agent also has a challenge in terms of security. In order to accomplish an operating agent system with mobile phones, we need to address and overcome those challenges. The mobile agent system has a great potential for practical applications from integrating of so called Internet of Things and wireless sensor networks into the Internet to the crowd sensing campaigns [17, 18]. We will refine our mobile agent system by integrating the challenges described in the previous section and implement practical applications to demonstrate its usefulness.

Acknowledgment This work is supported in part by Japan Society for Promotion of Science (JSPS), with the basic research program (C) (No. 25350482), Grant-in-Aid for Scientific Research.

References

1. Abe, T., Takimoto, M., Kambayashi, Y.: Searching targets using mobile agents in a large scale multi-robot environments. In: Proceedings of Agent and Multi-agent Systems: Technologies and Application. LNCS, Vol. 6682, pp. 211–220. Springer, Berlin (2011)
2. Sugiyama, S., Yamachi, H., Takimoto, M., Kambayashi, Y.: Aggregating multiple robots with serialization. In: Proceedings of Intelligent information and Database System. LNCS, Vol. 7196, pp. 177–186. Springer, Berlin (2012)
3. Aviles, A., Takimoto, M., Kambayashi, Y.: Distributed evacuation route planning using mobile agents. Trans. Comput. Collect. Intell. **XVII**, 128–144 (2014). (LNCS 8790)
4. Kambayashi, Y., Harada, Y.: A Resource Discovery Method Based on Multi-agents in P2P Systems. Intelligent Agents in the Evolution of Web and Applications. SCI 167, pp. 113–135. Springer, Berlin (2009)
5. Meier, T., Dunkel, J., Kakuda Y., Ohta, T.: Mobile agents for service discovery in ad hoc networks. In: Proceedings of 22nd IEEE International Conference on Advanced Information Networking and Applications, pp. 114–121. IEEE Press, New York (2008)
6. Yatsuyanagi, N., Takimoto, M., Kambayashi, Y.: Design of a return route support system using multiple mobile agents. In: Proceedings of SICE Annual Conference, pp. 1571–1576 (2014)
7. NHK Safety Information in Time of Disaster, http://www3.nhk.or.jp/anpi/
8. Askul Safetylink24, http://www.askul.co.jp/f/special/anpi/safetylink24/
9. Nishiyama, H., Ito, M., Kato, N.: Relay-by-smartphone: realizing multi-hop device-to-device communications. IEEE Commun. Mag. **52**(4), 56–65 (2014)
10. Fall, K.: A delay-tolerant network architecture for challenged internets. In: Proceedings of 2003 Conference on Applications, Technologies, Architectures and Protocols for Computer, pp. 27–34 (2003)
11. Shibata, K., Takimoto, M., Kambayashi, Y.: Expanding the control scope of cooperative multiple robots. In: Proceedings of 8th KES International Conference on Agent and Multi-agent Systems: Technologies and Applications, AISC, Vol. 296, pp. 17–26. Springer, Berlin (2014)
12. Comer, D.E.: Computer Networks and Internets, 6th edn. Pearson Education, New York (2015)

13. Satoh, I.: MobileSpaces: a framework for building adaptive distributed applications using a hierarchical mobile agent system. In: Proceedings og IEEE International Conference on Distributed Computing Systems, pp. 161–168. IEEE Computer Society, New York (2000)
14. Kawaguchi, N., Toyama, K., Inagaki, Y.: MAGNET: ad hoc network system based on mobile agents. Comput. Commun. **23**(8), 761–768 (2000)
15. Johnson, D.B., Maltz, D.A., Broch, J.: DSR: The Dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins, C.E. (ed.) Ad Hoc Networking, pp. 139–172. Addison-Wesley: Boston (2001)
16. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. In: Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications, pp. 90–100 (1999)
17. Leppänen, T., Liu, M., Harjula, E., Ramalingam, A., Ylioja, J., Närhi, P., Riekki, J., Ojala, T.: Mobile agents for integration of internet of things and wireless sensor networks. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 14–21 (2013)
18. Ma, H., Zhao, D., Yuan, P.: Opportunities in mobile crowd sensing. IEEE Commun. Mag. **52** (8), 29–35 (2014)

Analysis and Characteristics of Automatic Reconfiguration Mechanisms in IoT Devices Network

Grzegorz Debita, Patryk Schauer, Mateusz Juźwiak, Jarosław Szumega, Artur Palka, Emil Barczyński and Patryk Pham Quoc

Abstract Article presents an example of hardware implementation of IoT device on Xilinx's Zynq SoC. Authors demonstrate architectural level solution for managing and reconfiguration of software application and programmable logic device. Article explains usage of SoC device as a hardware platform for generic, fast and easily reconfigurable IoT device implementation.

Keywords Internet of Things · System on Chip · IoT · SoC · Reconfiguration · Software update · Software actualization · IoT device model

1 Introduction

Internet of Things (IoT) is a name of a new way of sharing information between objects in modern teleinformatics [1–4].

There is huge amount of papers on its architecture, uses and protocols, but there are not many researches considering hardware realization of IoT devices [1–4]. It is worth noting, that dynamic progress in digital technologies together with advancements in integrated circuits allows us to design IoT devices in line with an SoC (System on Chip) idea. Its point is to put all hardware functionalities in one integrated circuit [5–7]. Modern SoCs often consists not only of a processor, but also from a programmable logics. Such solutions are implemented e.g. in systems named Zynq or Kintex, produced by Xilinx company. Presented systems can be

G. Debita (✉) · P. Schauer · M. Juźwiak · J. Szumega · A. Palka
E. Barczyński · P.P. Quoc
Wrocław University of Technology, Wyb. St. Wyspiańskiego St. 27,
50-390 Wrocław, Poland
e-mail: grzegorz.debita@pwr.edu.pl

P. Schauer
e-mail: patryk.schauer@pwr.edu.pl

M. Juźwiak
e-mail: juzwiakmateusz@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_17

successfully employed in creation of hardware applications for processing of digital signals. In SoCs there is a possibility of hardware implementation of modulators and physical layer coders in radiocommunication and radiodiffusive systems [8–10]. SoCs using processors and Field-programmable gate arrays (FPGA) have practically limitless capabilities in a field of designing and prototyping new devices for IoT systems. Implementation of presented features can be also done using SoC systems of other manufacturers, such as Altera company. One of their implementations is Cyclone V system. This article presents main problems of reconfiguration in IoT system consisting of SoC type devices. Architecture of system governing reconfigurations and method for conducting them are also presented. Zynq system of Xilinx company was used as an example, due to considerable experience in working with this system of research team.

2 SoC Architecture in Relation to IoT Devices

General architecture outline of SoC has been presented in paper [1]. Device consists of two types of processors: System processor, which can be a unit in RISC architecture (Reduced Instruction Set Computing) and Dedicated Processor. System processor can be realized i.e. by a system of ARM Cortex family. Dedicated Processor has many possible realizations. For example it can be a dedicated processor used in Digital Signal Processing (DSP) or software type processor like LEON3 [11], Microblaze [12], Picoblaze [13] etc. It is assumed, that there is no limited amount of system and dedicated processors in SoC and it is dependant on system capacity and its application. Processors can be linked to communication interfaces. Presented SoC architecture uses its own operating memory and has an access to FPGA. Programmable arrays task is to support computations, which would take too many CPU time on processor. For example they would be computational accelerators for dedicated solutions like digital filters, modulators, demodulators and other algorithms processing signals and images, which implementation on hardware would help SoC application. System processor tasks are connected to managing of operating system. It is assumed, that in this case it would high-level operating system of Linux family.

That architecture can be related to proposition of authors of research [1], connected with IoT device. Authors of paper [1] suggests a model of device consisting of many communication interfaces, which task is to process signals from sensors and actuators as an online service. Suggested functionalities of IoT device can be easily implemented in SoC architecture.

Suggested concept of architecture can be fully implemented using SoC type devices. SoC devices equipped with programmable logic and processing systems have almost infinite potential when it comes to creating and reconfigurating structure of hardware realizing intelligent IoT devices. Examples of such application and implementation are shown in papers [14] and [15].

Moreover this kind of IoT hardware implementation allows to build universal devices. Generic communication subsystems have to be already prepared, but the rest of protocols stack and even modulation types (in case of wireless communication) can be programmed later. Devices functionality is limited only by number and type of connected sensors.

3 Architecture of IoT in Relation to SoC Devices

In scientific literature [18–21] architecture of IoT system is described using referencing model. Referencing model presented in paper [16] shows only outlines possible applications for IoT systems. For the sake of the model some terms were defined, such as "Virtual device", "IoT device", "Application". Details are shown on Fig. 1. Suggested model does not define which types of devices should be used to build an IoT application. Authors of paper [18] suggest using M2M (machine-machine) communication to realize communication between devices in an IoT network [17]. It is not hard to see, that IoT architecture presented on Fig. 1 is not a typical layer model like ISO/OSI model. It is rather an set of simultaneously cooperating services providing the work of the device and supporting the application, which works on it. Above model can be viewed as a general suggestion for an IoT system.

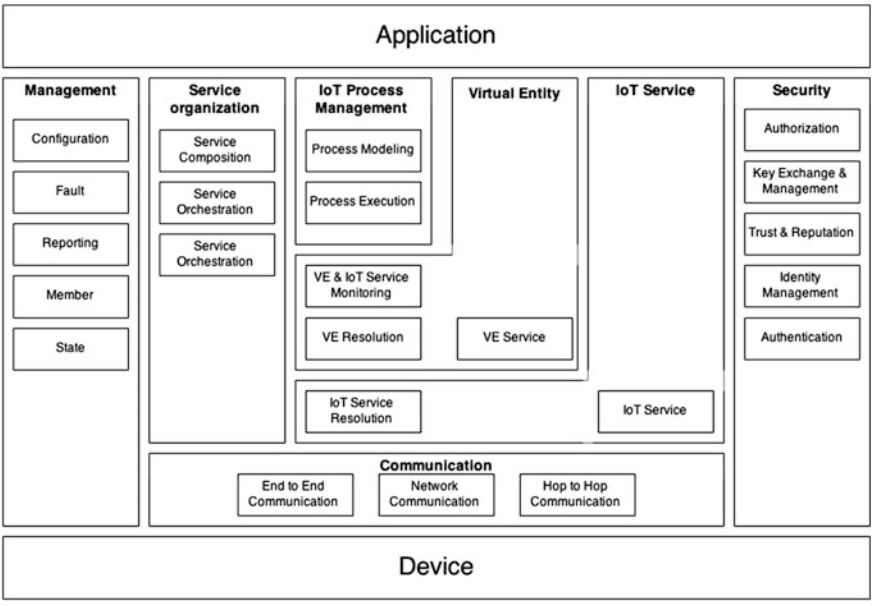

Fig. 1 Referencing IoT model based on [16] and [18]

When analyzing the above architecture model, it can be seen, that SoC devices from the 2nd chapter are ideal examples of IoT devices. Wide array of possibilities, which are provided by hardware reconfiguration lead to conclusion, that intelligent devices will be a dominant force in IoT networks.

Additionally simplicity of functionality reconfiguration allows the generic device to serve multiple service on demand. Previously installed device equipped with generic communication stack and multiple set of sensors, actuators and hardware links to other systems once can be utilized as monitoring station, otherwise can be processing node or even actuator controller. Moreover all services potentially provided by device can be register and published to users. Device, for example it is FPGA part, would be reprogramed on demand when the request arrived.

4 Management Mechanism Supporting Reconfiguration

4.1 Architecture of Reconfiguration

Most IoT devices currently present on the market has fixed functionality. Usually they are electronic systems based on microprocessors [20]. Their physical structure limits options for hardware improvement. Due to technological limitations issue of reconfiguration of devices does not get much attention lately. Devices in IoT networks are meant to fulfill specialized tasks. The case of SoC devices presented in this article is different, however, as their flexible internal structure, which can be freely reconfigurated and adjusted to used peripheral devices, is their main advantage [21].

There are Continuous Integration systems on market, such as Jenkins, although they were meant strictly for software, especially for web or mobile applications. They were not designed with IoT platforms in mind. Continuous Integration system for IoT platform (and a specific case of SoC devices), must fulfill several requirements and face problems, which are not present in other systems.

The possibility of reconfiguration requires to prepare a special architecture managing software updates. Dedicated services are needed both for target IoT device and for server side. There are many factors to be considered during designing of reconfiguration system, such as:

- permissible unavailability time (such as: actualization time, system booting time),
- system restore in case of actualization/reconfiguration failure,
- frequency of updates and bandwidth usage,
- backward compatibility of devices, which have not been reconfigured yet,
- providing quality of services during actualizations (server load during actualizations).

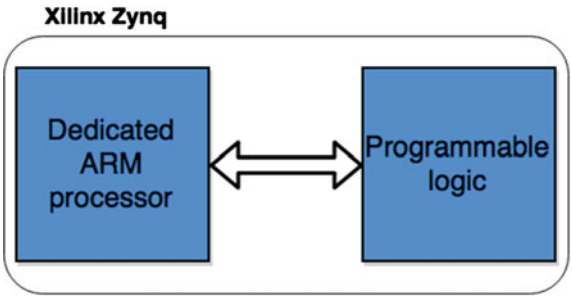

Fig. 2 Xilinx Zynq—simplified diagram of internal communication

Structure of Zynq platform (Fig. 2) and its capability of communication between logical part and operational system allows to create reconfiguration mechanisms implemented next to logical structure. Putting all needed functions in a one device is a very convenient solution.

If there is a will to use reconfigurable platform as a basis for IoT device, following mechanisms should become a main focus:

- Server-side—managing of a given part of devices—establishing an actual version of firmware, giving proper orders of e.g. actualization, monitoring of basic parameters
- Client-side—device managing—connecting to server, downloading proper information and files, correct autoconfiguration with a possibility of reverting to last working version of firmware.

4.2 Server-side Reconfiguration Mechanisms

Reconfiguration of software is required in a case of:

– changing an essential parts of firmware or software,
– changing configuration of selected device,
– changing hardware devices connected to device.

Due to that fact, mechanisms supporting reconfiguration should be synchronized with the software development cycle (Fig. 3). A very convenient solution is to link them with version control system, such as a popular GitHub. Every software modification stored in a system ("commited") can be treated in two ways—as a part of greater functionality or as a ready next version of software. In the first case, programmers aim is only to store history and versions of their work.

The second case (adding new functionality and, in effect, preparing the next version) should be preceded by tests verifying the program's correctness. Such tests should be conducted on virtual device, which could be realized by a service (if one of many functions of the same software is developed, i.e. in a form of plugin) or a test device, being an equivalent of an IoT device, which the new firmware will be

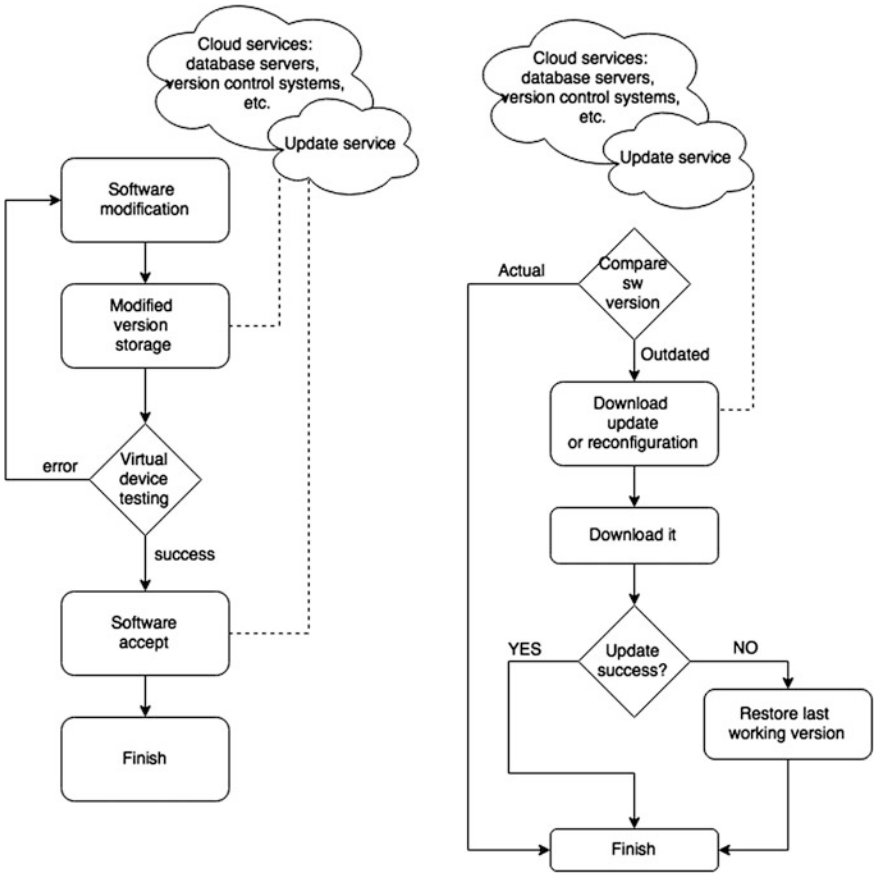

Fig. 3 Flowchart of software development cycle and software update on an IoT device

working on. After successful tests there should be a possibility of accepting given version to be used on target devices. Then the actualization would take place on dedicated server. To identify new software server should give it an unique ID number, e.g. a commit identifier.

Additionaly a message about changing the version should be sent to all devices or the device itself should query server about it in a fixed intervals.

4.3 Reconfiguration of Target Device

Suggested solution assumes creating the network of target devices, which will communicate with the server, on which they will be able to find available software (especially firmware) and actualizations. In case of outdated software, device

should download and install new version. Depending on the needs, process of installation can also contain compilation of software.

If operation ends successfully and work of all functions is confirmed, the actualization process will end. In case of failure it is necessary to restore the previous working version of software. Automatic procedure is huge advantage, because a break in system functioning is minimal. Flowchart of this process is presented on Fig. 3.

4.4 Architecture of Test Network

According to the reconfiguration algorithm presented on Fig. 3 the test architecture was defined, which allows to verify efficiency and correctness of suggested solution.

A server, which will share the version control services and store actualizations and configurations, is necessary. At SoC device, actualization can be understood in several ways. It can be:

- application software working under the operational system,
- operating system,
- software for dedicated processors,
- bitstream file for FPGA system,
- configuration for operating system,
- configuration for application software,
- configuration for dedicated processor/FPGA system.

Every of the above mentioned types of actualization requires a dedicated software, which would:

- check the availability of actualization,
- download an actualization,
- install actualization.

Those processes can be executed by separate applications or by one.

Procedure of installing actualization will be different, depending on which device the actualization is designed for. Each actualization type is conducted in a different way and usually requires different software. In case of FPGA system programming another problems and requirements can be present, such as need for partial system reconfiguration in order to maintain availability of services or partial reconfiguration as a result of universality of prepared actualizations. In such case an actualization configuration file should come together with an actualization.

It should be noted, that software updaters applications can also be changed. Such situation can take place i.e. when the device profile is completely changed or as a result of design considerations connected to technology of developed software. Due to that, in certain applications it can be necessary to prepare mechanisms updating

actualizing software. Implementation-wise it can be a mechanism updating the whole FLASH memory on a bootloader level.

Feedback about update success is another issue. Server should have an information about connected devices and their versions. In case of update failure it should get a proper message together with description of error. An error can occur due to wrong software, configuration or applying an update to a wrong device. System operator should know about every abnormality of actualizations, especially if their wrong configuration makes a proper functioning of target device impossible.

Server ought to automatically share actualization set designed for given device in that way, to ensure compatibility of used software versions. It can be clearly seen, that there is a need for storing actualization and reconfiguration revisions.

Additionally, in a constant-working systems, especially high-availability systems, updates should be made by a strict schedule. Server which manages updates should send actualizations to devices in the very moment there were intended to. In case of failure immediate measure should be taken to solve the problem.

It is a good idea to make update server independent of server providing services to application. It will increase system's reliability in case one of them is unavailable.

It is necessary to make important decisions concerning system's architecture during design step. A system managing the reconfiguration of devices has to be an integral part of IoT architecture, because it is critical for its development. Aforementioned mechanisms allow to:

- achieve maximum availability of the system,
- eliminate the option of operator's failure due to providing a set of compatible software versions and configurations,
- systematically provide new software versions for devices—even during product development cycle.

Due to its complexity, designing the update management system is not an easy task. Preparing the system for a vast array of architectures and IoT devices requires consideration of many factors. Versatile architecture would allow to solve many problems occurring in present systems.

Reconfiguration of FPGA system in presented solution, together with software updates, gives many options. It allows to add new functionalities, i.e. support of new device, devices connected by the consecutive communication interfaces—all this without a break in services availability. It is ensured by the partial reconfiguration mechanism of the FPGA system. Currently used part of the system is not changed and works constantly—only the idle part is modified. New configuration can contain new functionalities, which are not tied in any way to those already present in system. It is a great advantage of SoC architecture in relation to IoT devices. Hardware limitations, limited number of communication interfaces and limitations of sequential data processing simply disappear. Dedicated DSP or FPGA systems allows to adjust computational and communicational capacities to the present needs and application.

4.5 IoT Multiservice Device

Device functionality update procedure was described in previous sections. But as it was mentioned above there is possible not only to upgrade software but also to change device functionality. Especially, device can potentially provide many different services, which could not be possible in the ordinary hardware implementation. SoC technology enables possibility of changing device functionality on demand.

This kind of IoT universal device utilization scenario bases on SOA (Service Oriented Architecture) paradigm. Services potentially provided by the device have to be registered. When one of this services are requested specific software is downloading from external source (network scenario) or from storage attached to device. Device operating system control process of reprogramming and configure requested service. After this process the device is ready for providing new functionality without any hardware reconfiguration. Moreover it can be completely autonomous process executed without any user knowledge. The only limitation is number of external devices like sensors, actuators and links to other systems.

5 Summary

In this article an example of reconfiguration method for an intelligent device made in SoC technology was presented. Demonstrated experiment was performed in a laboratory conditions with a help of evaluation kit called ZedBoard, made by Digilent company. The kit was equipped with an additional interface card FMCOMMS3 by Analog Devices. Our experimental aim was to conduct remote reconfiguration of FPGA system and software needed to manage installed hardware. Those tests were successful, which proves that remote reconfiguration of intelligent devices is possible. Presented case is only one of many possible uses of such solution. The authors showed an example of SoC device based on SDR technology, due to their experience with implementation of hardware solutions in radiodiffusive systems. It does not change the fact, that the solution is highly versatile and ultimately will be used in another IoT applications, i.e. intelligent sensor systems, intelligent drivers for motorization and transport systems, intelligent devices for monitoring systems used by civil and military services.

The authors assumed next steps to be done in the future. They consist of designing an automatic reconfiguration system for remote group of devices in M2M communication, which would have an access to the Internet; creating a safe communication protocol for hardware and software reconfiguration procedures on intelligent device; conducting lab experiments for a larger amount of devices. Results of that scientific research will be published in next articles.

Acknowledgments The research presented in this paper was partially supported by the Polish Ministry of Science and Higher Education and the European Union within the European Regional Development Fund, Grant No. POIG.01.03.01-02-079/12 and within European Social Fund.

References

1. Schauer, P., Debita, G.: Internet of things service systems architecture. In: New Trends in Intelligent Information and Database Systems, pp. 239–248. Springer International Publishing, Berlin (2015)
2. Miorandi, D., Sicari, S., Pellegrini, F.D., Chlamtac, I.: Internet of things: vision, applications and research challenges. Ad Hoc Netw. **10**(7), 1497–1516 (2012)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. pp. 2787–2805 (2010)
4. Bonetto, R., et al.: Secure communication for smart IoT objects: protocol stacks, use cases and practical examples. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012, pp. 1–7. IEEE, NJ (2012)
5. Keating, M., Flynn, D., Aitken, R., Gibbons, A., Shi, K.: Low Power Methodology Manual: For System-on-chip Design. Springer Publishing Company, Incorporated, Berlin (2007)
6. Lackey, D.E., Zuchowski, P.S., Bednar, T.R., Stout, D.W., Gould, S.W., Cohn, J.M.: Managing power and performance for system-on-chip designs using voltage islands. In: Computer Aided Design (2002)
7. Hwang, L.J., Sanchez, R.L.: U.S. Patent No. 7,058,921. Washington, DC: U.S. Patent and Trademark Office (2006)
8. Debita, G., Szumega, J., Palka, A.: Techniki odbioru i monitorowania niskobudżetowych stacji nadawczych DAB/DAB+/DMB zbudowanych za pomocą narzędzi opensource, Przegląd Telekomunikacyjny, Wiadomości Telekomunikacyjne, R. 83, nr 8/9, s. 1078–1082 (2014)
9. Debita, G., Barczyński, E., Pham Quoc, P.: Prototypowy system zarządzania siecią niskobudżetowych stacji nadawczych radiofonii cyfrowej DAB/DAB+/DMB nadających w SFN, Przegląd Telekomunikacyjny, Wiadomości Telekomunikacyjne, R. 83, nr 8/9, s. 1073–1077 (2014)
10. Debita, G., Penar, W., Greblicki, J., Ostrowski, M.: Niskobudżetowa stacja nadawcza DAB/DAB+/DMB zbudowana za pomocą narzędzi i oprogramowania open source. Przegląd Telekomunikacyjny, Wiadomości Telekomunikacyjne, R. 87, nr 4 (2014)
11. Gaisler, J., Catovic, E., Multi-core processor based on leon3-ft ip core (leon3-ft-mp). In: DASIA 2006-Data Systems in Aerospace, vol 630, p. 76 (2006)
12. Hubner, M., Paulsson, K., Becker, J.: Parallel and flexible multiprocessor system-on-chip for adaptive automotive applications based on xilinx microblaze soft-cores. In: Proceedings of 19th IEEE International Symposium on Parallel and Distributed Processing, pp. 149a–149a. IEEE, NJ (2005)
13. Chapman, K., PicoBlaze 8-bit microcontroller for virtex-E and spartan-II/IIE devices. Xilinx Appl. Notes (2003)
14. Pfeifer, P., Pliva, Z.: On measurement of parameters of programmable microelectronic nanostructures under accelerating extreme conditions (Xilinx 28 nm XC7Z020 Zynq FPGA). In: Proceedings of 23rd International Conference on Field Programmable Logic and Applications (FPL), 2013, pp. 1–4. IEEE, NJ (2013)
15. Korcyl, G., Moskal, P., Bednarski, T., Białas, P., Czerwiński, E., Kapłon, Ł., Zoń, N.: Trigger-less and reconfigurable data acquisition system for positron emission tomography. Bio-Algorithms Med-Syst. **10**(1), 37–40 (2014)
16. Haler, S.: IoT-i deliverable D1.5: IoT Reference model white paper, Sept. 2012

17. ETSI TS 102 690: Machine-to-machine communications (M2M), Functional architecture (2011)
18. Krco, S., Pokric, B., Carrez, F.: Designing IoT architecture (s): a European perspective. In: IEEE World Forum on Internet of Things (WF-IoT), pp. 79–84. IEEE, NJ (2014)
19. Bassi, A., Lange, S.: The need for a common ground for the IoT: the history and reasoning behind the IoT-a Project. In: Enabling Things to Talk, pp. 13–16. Springer, Berlin (2013)
20. Limin, H.: Internet of things part IV: embedded systems in age of IOT. Microcontrollers Embed. Syst. **1**, 034 (2012)
21. Lysaght, P., Blodget, B., Young, J., Bridgford, B.: Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs. In: Proceedings of FPL (2006)

Cross-Population Semiosis in Multi-agent Systems

Wojciech Lorkiewicz and Radosław Katarzyniak

Abstract Semiosis mechanism in an artificial system should guarantee the development of a consistent and common substances of language symbols. Thus allowing a population of interacting agents to autonomously learn, adapt and optimise their semantics. In this research we define a settings for the cross-population semiosis model, which involves two or more mature populations set together in a common environment with a goal to align their predefined (or differently developed) lexicons. In particular, using the language game model we introduce a new type of language game scenario and define a set of specific measures that capture the dynamics of lexicon evolution in cross-population semiosis model. Finally we provide an experimental verification of the behaviour of the alignment process of cross-population semiosis, as such test the applicability of the classical language game model approach.

Keywords Agent · Language game · Semiosis process · Cognitive semantics

1 Introduction

Linguistic communication requires from individual agents to have access to a predefined and shared set of proper names (for all of available phenomena, for instance for all objects in the environment). In essence, appropriate communication mechanism requires that all interacting agents use a well-defined and shared set of individual names for objects about which the agents can, and intend to, linguistically refer to. For instance, in order to efficiently attract the attention of an agent to a

W. Lorkiewicz (✉) · R. Katarzyniak
Faculty of Computer Science and Management, Wrocław University of Technology, Wrocław, Poland
e-mail: wojciech.lorkiewicz@pwr.edu.pl

R. Katarzyniak
e-mail: radoslaw.katarzyniak@pwr.edu.pl

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_18

particular object present in the current state of the environment an individual should utilise a shared name.

Unfortunately, populations are often located in partially (or even fully) unknown environments. Therefore it becomes impractical to assume that a complete lexicon (word-object mapping) can be developed for and shared by populations at the design time. As such, it is unpractical to assume that the required names can be defined and shared beforehand.

In consequence, it is rational to equip individual agents with means to develop their own lexicons, where the development is understood as the process of introducing novel word-object pairings stored in the lexicon as well as aligning the already existing ones. In this paper we model this ability as an explicit internal learning mechanism that, through a series of consecutive interactions between the agents, can lead to the alignment of individual lexicons, and ultimately to the formulation of a coherent and shared lexicon (naming convention).

The aforementioned task relates directly to the problem of language alignment in an artificial population of autonomous knowledge units. Due to its pragmatic nature this problem is fundamental to the field of multi agent systems, especially in case of embodied multi agent systems. For instance, it has been shown that incorporating a flexible semantic communication system into a smart sensors network (See [1–3]) may lower the system's energy utilisation and inevitable extend its operation time; similarly in the case of language alignment in robotic systems (See [4, 5]); etc.

In order to develop a shared lexicon an agent should be able to introduce new names and be able to align its naming convention with the rest of the population. On the one hand, the naming convention is build up by agents from scratch, i.e., whenever an agent encounters a novel object it is able to associate it with a linguistic label—word-object pairing. Whereas on the other hand, the entire population must agree on a certain set of pairings, i.e., the agent is able to modify its internal associations according to the population need of consistency. In particular, through a series of consecutive interactions between the agents certain word-object associations should become more and more aligned, finally reaching a shared, unified naming convention.

In this research we evaluate the proposed alignment process and study the dynamic character of the formation of coherent naming conventions. Following the language game model approach (See [6, 7]), we provide an intuitive, yet still in depth, example of name alignment process in a novel scenario comprised of multiple interacting populations. In particular, using a simulated multi agent system we give insights on the effects of different population structures in the case of the least restrictive type of naming game (without feedback). We show how a change in a concentration of interaction, cross- or inner-population, may affect the overall convergence of alignment processes and the overall demand on the memory. In particular, the straightforward incorporation of a population semiosis mechanism in such a setting might be inadequate. Mainly, due to the fact that it is necessary for the populations to sustain a relatively high level of communication efficiency within original populations (inner-population interaction).

2 Semiosis in Multi-agent Systems

Semiosis is the process of shaping the meaning of language symbols in a given population. The term semiosis (coined by C.S. Pierce) describes the process that interprets signs as referring to their real world objects. As noted by de Saussure (See [8]), the meaning of a symbol in a given population results from a certain convention and is a result of a common agreement—a sign cannot function until the audience distinguishes it. As such the language, in general, is strictly correlated with the process of conceptualization that depends on the environment, empirical experience, and internal organization of an individual. In practice, every language symbol should result from the process of semiosis, as only in such case it is convenient to introduce the notion of shared understanding of language symbols among agent's population.

In our previous work (See [9–11]) we focused on two basic scenarios of the semiosis process in autonomous systems, namely individual semiosis and population semiosis. The former focuses on the processes of language alignment, where an individual agent is learning a proper word-object mappings from a mature language user—a teacher. Whereas, the latter focuses on the process where multiple individuals thrive to establish a common lexicon without the presence of an established and proficient language user. In this case the agents not only need to align their individual lexicons (with other agents), but also need to introduce novel language symbols for unnamed objects (perceived in the environment).

In this paper we focus on the third scenario, in which the individual agents are arbitrary arranged into groups (populations) and pre-equipped with mature lexicons (defined word-object mappings within consistent groups of agents), and thrive to develop a shared (cross-group) word-object mappings (See Sect. 2.2).

2.1 Language Game Model

In principle, developing an alignment mechanism that would lead to a coherent formulation of names among interacting individuals is not a trivial task. Several approaches have been proposed and investigated in the literature (See [4, 8, 12–14]), ranging from associative types of memory (See [6, 8]), through genetic algorithm models (See [14]), to neural network adaptations (See [4]).

Despite the large diversity of different approaches the Language Game Model (LGM for short) (See [7, 8]) seems to be the most pragmatic, popular and widely applied solution. The major advantage of the language game model lies in a simple interaction routine, associative meaning-word representation and cross-situational learning mechanism. Moreover, it has been shown that in certain environment settings such a mixture leads a population of interacting agents to a coherent naming convention.

In the LGM the interaction between the agents is governed through a series of consecutive basic individual episodes. Each such episode involves two randomly selected agents. Both are situated in the same state of an external environment (common context), where one is acting as a speaker, whilst the other as a hearer. In such a setting the speaker tries to draw attention of the hearer to a particular element in a shared scene (intended topic), using a simple linguistic clue (one word utterance). Further, the hearer tries to correctly interpret the linguistic utterance as a particular element of the current state of the external environment (the hearer has very limited information[1]).

The essence of the idea incorporated in the LGM is hidden in applying the hearer's internal linguistic processes over multiple learning episodes. Consequently, after receiving enough samples the agent is able to build adequate correlations between utilised linguistic symbols and observed states of the external environment.

The entire model is built on top of the assumption that the agents share a common environment (though only a part of the environment is available at a given point of time) and share a common communication channel (here a linguistic communication, where the agents exchange objects' names). This common ground allows the agents to activate particular instances of the semiotic triangle—grounding the language symbols.

Unfortunately, the alignment process in the LGM is strictly individual. First, each agent is managing its own, strictly private, lexicon. Second, it is each autonomous agent that introduces changes to the linguistic structure of the lexicon (though based on the interaction with other agents). Yet there is no explicit notion of the linguistic stance of the entire population, and there are no means to determine the direct success of the individual communication, nor there are no means for the individual agent to identify its interlocutor. As such performing an experimental verification of the applicability of such an approach to a more complex case of semiosis seems justified and valuable.

2.2 *Cross-Population Scenario*

The cross-population case of semiosis (See Fig. 1) involves two or more mature populations[2] (maturity of the population is defined in terms of their linguistic capabilities), each having differently established semantics (predefined or differently developed as of the process of population semiosis (See [11]). These

[1]It should be underlined that in the assumed type of language game the agents do not receive any form of direct feedback concerning the outcomes of the game. As such the interpreted meaning and the heard word are solely regarded as the most probable ones.

[2]In this paper we limit our research to a case of co-existence of two mature populations (depicted by A and B). Despite the fact that this is an obvious simplification, the obtain results can be generalised over more complex scenarios.

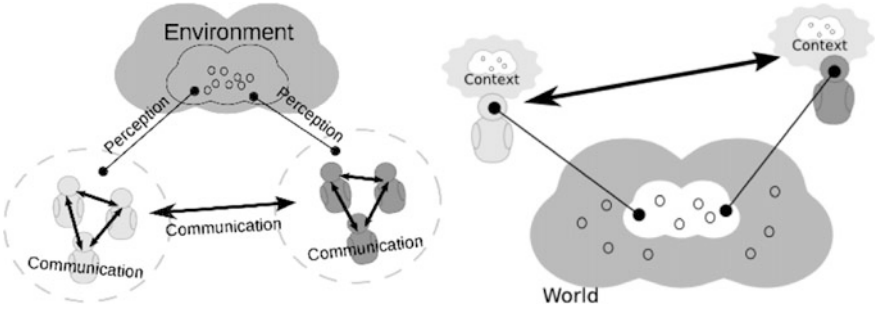

Fig. 1 Cross-population semiosis model

populations are set together in a common environment where they can interact with each other.

Treated as a whole the collated populations may have various meanings assigned to the same language symbols. In such a case in order to allow cross-population communication their internal lexicons (semantics) should be aligned.

Additionally, different populations may use different language symbols for the same objects. In such a case in order to allow cross-population communication both dictionaries should be related to each other and lexicons aligned.

However, as the aligned populations (as well as their languages itself) share the same environment, common to all populations, the environment serves as a common point of reference. As the existence and state of the external world is objective (rather than subjective) the agents are able to establish a proper correlation between different original languages.

As opposed to the traditional case of language alignment, here two major interaction types can occur:

- Internal—the agents from the same population can engage in an interaction (See Fig. 2), thus enforcing the strength of local lexicons. In particular, both interacting agents (speaker and hearer) are from the same population (either A or B).
- Cross-population—the agents from different populations can engage in an interaction (See Fig. 2), thus weakening the strength of local lexicons. In particular, interacting agents come from different populations (either speaker from A and hearer from B, or speaker from B and hearer from A).

The ultimate goal of the alignment process in the cross-population semiosis scenario is again to develop a shared and common substance of symbols that are shared among all of the interacting agents (agents from all populations). However, the quality of the resultant lexicon can be rated not only by the rate of its commonality among the agents, but also by the distance to the original inner population lexicons. Ideal realisation would require that the resultant lexicon is shared among all of the interacting agents, and that its distance to all original inner population lexicons is minimal. As such it is important to take into consideration these two factors in the evaluation process.

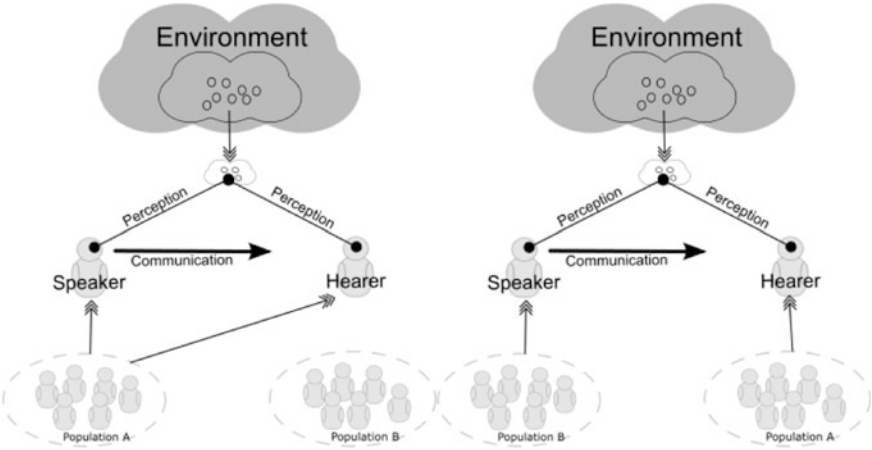

Fig. 2 Cross-population example—internal communication (*left*), cross communication (*right*)

2.3 Cross-Population Model

Realisation of the cross-population semiosis requires a mixture of two types of interaction (internal and cross), where both the cross population and inner population communication is present in the system. In particular, such an approach allows the agents to develop a shared language between the populations and maintain the internal consistency within the original populations.

Due to the co-existence of multiple populations, we need to extend the basic notion of a system state (as defined for a single population model, See [10, 11]) in the cross-population semiosis model. The state of the entire multi-agent system is defined as a tuple S_t^{C-P}, as follows:

Definition 1 (*Cross-population Semiosis System*):
For each $\mathrm{t} \in \mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots\}$ a system state S_t^{C-P} in time point t is a tuple:

$$S_t^{C-P} = \langle O, P, W_t, X_t^O, X_t^P \rangle, \tag{1}$$

where:

- $O = \{o_1, o_2, \ldots, o_M\}$ denotes the set of fixed and static objects,
- $W_t = \{w_1, w_2, \ldots, w_{R(t)}\}$ denotes the set of all words available in the system at time point $t \in T$

- $\mathrm{P} = \mathrm{P}_1 \cup \mathrm{P}_2 \cup \ldots \cup P_N$ denotes the population divided into multiple disjoint internal populations $P_i = \{a_1^i, a_2^i, \ldots, a_{Ni}^i\}$ (a_k^i denotes a single agent from population i),[3]
- $X_t^O{:}T \rightarrow 2^O$ denotes the context of interaction at time point $t \in T$
- $X_t^P{:}T \rightarrow 2^P$ denotes the pair of interacting agents at time point $t \in T$, the value $X^P(t) = (a_k^i, a_l^j)$ is a pair, where a_k^i denotes a speaking agent and a_l^j denotes the hearer.

Due to the existence of multiple population there is a need to introduce two types of interactions: inner-population ($\mathrm{X}_\mathrm{t}^{\mathrm{P,inner}}$) and cross-population ($\mathrm{X}_\mathrm{t}^{\mathrm{P,cross}}$). The former involves the interplay between agents originating from the same population, whilst the latter involves the interplay between agents originating from different populations.

Definition 2 (*Inner-/Cross-population Interaction*):
The agents' inner- and cross-interaction, realized at a given time point $t \in T$, are defined by the interaction functions as follows:

$$X_t^{P,inner}{:}T \rightarrow 2^{P_i} \times 2^{P_i} \tag{2}$$

$$X_t^{P,cross}{:}T \rightarrow 2^{P_i} \times 2^{P_j}, \quad i \neq j \tag{3}$$

To manage the two types of interaction we introduce the notion of inner interaction probability. It defines the frequency of inner-interactions to all interactions that take place in the system. As such, this parameter governs the tendency to maintain close relationships within the original populations.

Definition 3 (*Inner Interaction Probability*):
Inner interaction probability $p_{inner}(i) \in [0, 1]$ for population i is defined as:

$$p_{inner}(i) = \Pr\left(X^P(t) = \left(a_k^i, a_l^i\right){:}a_k^i, a_l^i \in P_i\right) \tag{4}$$

Agent's internal state is used to represent the state of agent's embodied linguistic subsystem, given at a particular time point $t \in T$.

Definition 4 (*Agent State*):
For each $t \in T = \{\mathrm{t}_1, \mathrm{t}_2, \ldots\}$ agent's a state S_t^a in time point t is a tuple:

$$S_t^a = \langle O_t^a, W_t^a, L_t^a, \Gamma_t^a, \delta_t^a, \theta_t^a \rangle, \tag{5}$$

[3]As aforementioned, we further assume that the system consists only of two interacting populations, i.e., P_1 and P_2.

where:

- $O_t^a \subseteq O$ denotes the set of all objects known (already encountered) by agent a,
- $W_t^a \subseteq W_t$ denotes the set of all words known (already encountered or invented) by agent a,
- $L_t^a : W_t^a \times O_t^a \rightarrow [0,1]$ denotes the lexicon mapping embodied in the agent and represents an actual correlation between objects and words by setting up an association strength for particular word-object pairs
- $\Gamma_t^a : W_t^a \rightarrow [0,1]$ denotes the agent's subjective notion of usability (word strength) at time point $t \in T$,
- $\delta_t^a : W_t^a \times L_t^a \times \Gamma_t^a \rightarrow O_t^a$ denotes the interpretation function (interpretation of a particular external utterance),
- $\theta_t^a : O_t^a \times L_t^a \times \Gamma_t^a \rightarrow W_t^a$ denotes the production function (selection the most adequate name for a given object).

The straight forward approach from the population semiosis scenario might be inadequate, as it is necessary for the populations to sustain a high level of efficiency in communicating within original populations. In order to study the behaviour of the system we need to introduce additional measures that incorporate the model of multiple populations. As such, we need to distinguish the coherence at a given point of time within the original populations ($\boldsymbol{\mu_{LC}^{inner}(i,t)}$) and cross population coherence ($\boldsymbol{\mu_{LC}^{cross}(t)}$), and further study the number of iterations required to reach a particular coherence level α (**cross** $\boldsymbol{\mu_{TLC}^{cross}(\alpha)}$ or **inner** $\boldsymbol{\mu_{TLC}^{inner}(\alpha)}$) and the minimum coherence rate reached (**cross** $\boldsymbol{\mu_{MLC}^{cross}}$ or **inner** $\boldsymbol{\mu_{MLC}^{inner}}$). Further we utilise the maximum number of used words (**cross** $\boldsymbol{\mu_{UW}^{cross}}$ or **inner** $\boldsymbol{\mu_{UW}^{inner}}$), the maximum number of total words (**cross** $\boldsymbol{\mu_{TW}^{cross}}$ or **inner** $\boldsymbol{\mu_{TW}^{inner}}$), and the similarity between the original lexicons and the established lexicons (**cross** $\boldsymbol{\mu_{LD}^{cross}}$ or **inner** $\boldsymbol{\mu_{LD}^{inner}}$; generalised identity function).

Definition 5 (*Inter/Cross Population Coherence*):

$$\boldsymbol{\mu_{LC}^{cross}(t)} = \langle I\left[\delta_t^a\left(\theta_t^A(o)\right) = o\right]\rangle_{\boldsymbol{a,A \in P, o \in O}}$$

$$\boldsymbol{\mu_{LC}^{inner}(i,t)} = \langle \mathbf{I}\left[\delta_t^a\left(\theta_t^A(o)\right) = o\right]\rangle_{\boldsymbol{a,A \in P_i, o \in O}}$$

Due to limited space we further omit formal definition of additional measures used in the simulations.[4]

[4]We refer a curios reader to our previous work—See [10, 11].

3 Simulation and Analysis

Let us focus on the major part of this research that is the experimental verification of applicability of the language game model approach to two population cross-population semiosis model. Experimental verification was carried out based on a vast set of direct agent-based simulations. It should be stressed that the utilised simulation framework follows the restrictive guidelines of ABM approach. In particular, we study the behaviour of the alignment process in two major settings. First, with a globally set identical inner interaction probabilities ($p_{inner}(1) = p_{inner}(2) \in [0, 1]$), though varied in experiments. Additionally, we verify the behaviour of the alignment process in case of small (single word-object pair) and large (multiple word-object pairs) differences between the initial lexicons. Second, with a fixed inner interaction probability for a population A ($p_{inner}(1) \in \{0.25, 0.5, 0.75\}$) and varied for the other population B ($p_{inner}(2) \in [0, 1]$).

The carried out procedure is rather straightforward. We set two groups of agents with two predefined mature lexicons. The maturity of the lexicon lies in fact that the strength of the word-object mappings and word strength is at its maximum value. Then we establish the interaction probabilities, both inner- and cross-population. Finally we initialise the language game process and keep track of the aforementioned measures—inner/cross coherence rate, number of used/total words, and lexicon differences.

For the sake of simplicity and tractability, we additionally assume that all of the interacting agents are capable of establishing joint attention scene and share the entire set of currently available objects (context). In essence, we follow the no-feedback naming game model with a well-established mechanism of interpretation and production (See [15, 16]), with pair-wise communication scheme with random topic selection (See [11, 17]), context size of 2 objects, and with the ICSL alignment mechanism (See [10, 11, 16]). All of the presented graphs represent an average behaviour over at least 100 runs. The baseline settings assume two population of equal size (5 agents; 10 agents in total), environment consisting of 5 objects, equal probability of speaking agent selection (between two populations).

Let us no focus on a simple case with equal inner interaction probabilities and small lexicon difference against different inner interaction probabilities (See Fig. 3).

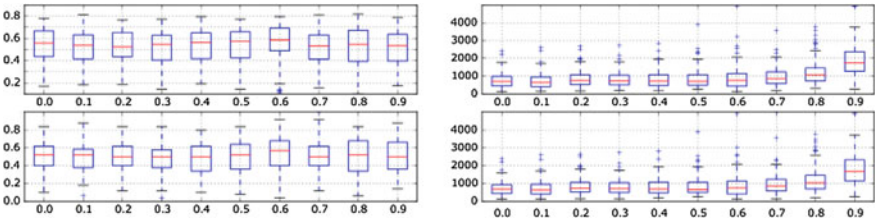

Fig. 3 μ^{cross}_{MLC} (*left*) and $\boldsymbol{\mu}^{\boldsymbol{cross}}_{\boldsymbol{TLC}}(\mathbf{0.99})$ (*right*) with equal inner interaction probabilities and small lexicon difference; cross-population (*top*), inner-population (*bottom*)

Obviously, in case of maximum inner interaction probability (IIP for short) $p_{inner}(1) = p_{inner}(2) = 1$ the system is unable to establish a coherent lexicon (never reaches coherence rate level of 0.99). Whereas, in all other cases the system is able to stabilise at a particular shared lexicon. Moreover, there seems to be no particular influence of different IIP on the minimum coherence rate, both inner- and cross-population wise. Yet, higher IIP values (tendency to maintain close inner-population relationships) directly influences the number of iterations required to reach a particular coherence rate (here 0.99). In practice, we can distinguish three areas of this influence: below 0.2, where the number of iterations is nearly constant at a relatively low level; between 0.2 and 0.6, where the number of iterations is again nearly constant but at a bit higher level; and above 0.6, where the number of iterations start to drastically increase with the increase of IIP. Such a behaviour can be easily explained, as low inner-population ties guarantee substantially more cross-population interactions which are actually the key to establishing a common (across groups) lexicon by degrading local tendencies. Whereas, high inner-population ties significantly limit such interactions and enforce the local internal structures. Interestingly, even in case where 9 out of 10 (IIP of 0.9) interactions are inner-population (thus highly enforce local lexicons) the population treated as a whole is able to reach a shared naming convention.

Further, nearly identical behaviour (naturally scaled proportionally to the amount of lexicon difference) can be observer in case of equal inner interaction probabilities and large lexicon differences against different inner interaction probabilities (See Fig. 4). In all cases (IIP different from 1) the system is able to stabilise at a particular shared lexicon, and again there seems to be no particular influence of different IIP on the minimum coherence rate, both inner- and cross-population wise. In practice, we can still distinguish three areas of the influence of higher IIP values on the number of iterations required to reach a particular coherence rate, though the difference between first two groups is less visible. Yet again, even in case where 9 out of 10 (IIP of 0.9) interactions are inner-population (thus highly enforce local lexicons) the population treated as a whole is able to reach a shared naming convention.

Next we can focus on a more complex case with non-equal inner interaction probabilities and small lexicon difference against different inner interaction probabilities (See Fig. 5). Non-equality of IIP is realised by fixing the value for one of

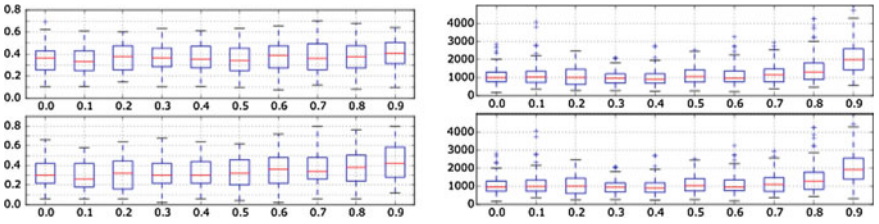

Fig. 4 μ_{MLC}^{cross} (*left*) and $\boldsymbol{\mu}_{\boldsymbol{TLC}}^{\boldsymbol{cross}}(\mathbf{0.99})$ (*right*) with equal inner interaction probabilities and large lexicon difference; cross-population (*top*), inner-population (*bottom*)

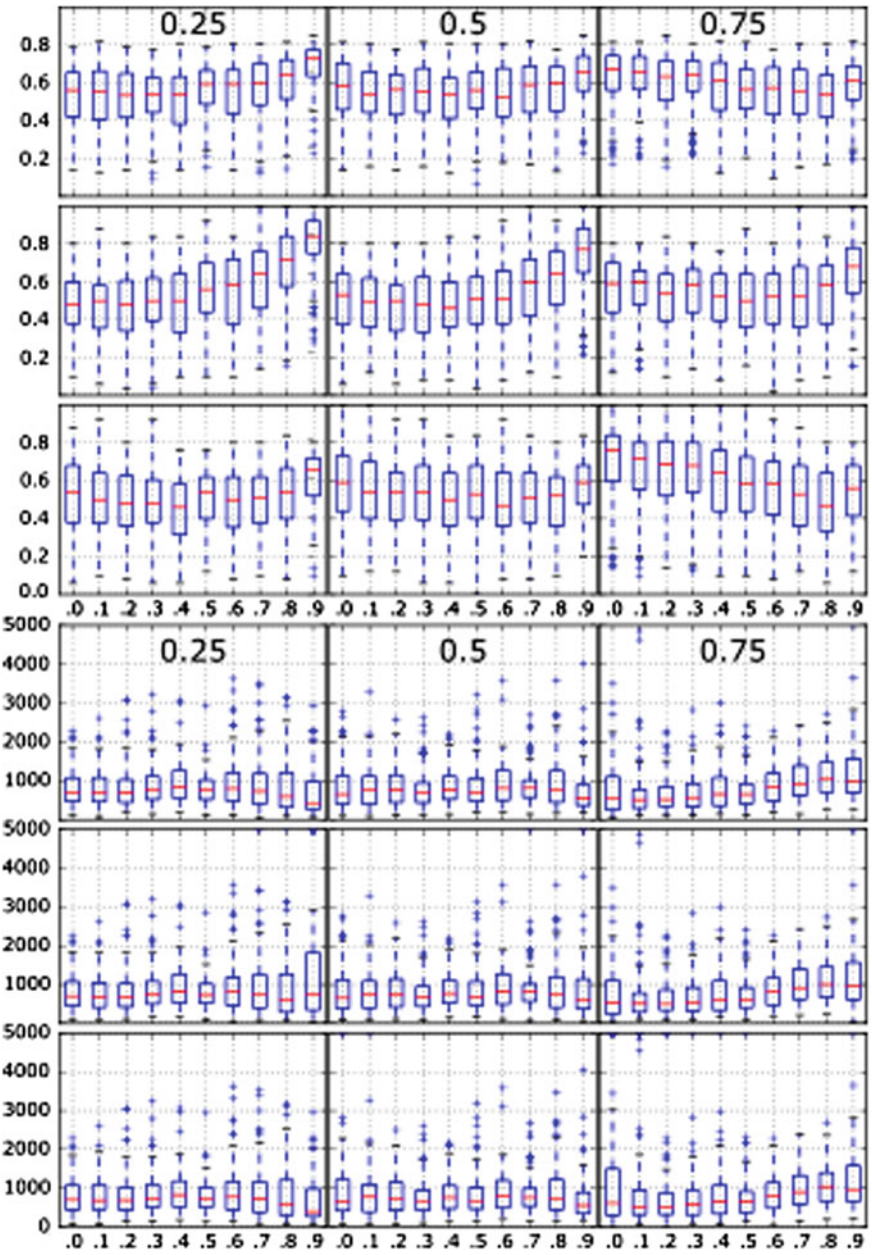

Fig. 5 μ_{MLC}^{cross} (*top*) and $\boldsymbol{\mu}_{\boldsymbol{TLC}}^{\boldsymbol{cross}}(\mathbf{0.99})$ (*bottom*) with non-equal inner interaction probabilities and small lexicon difference; IIP of 0.25 (*left*), 0.5 (*middle*), 0.75 (*right*); cross-population (*inner top*), inner-population A (*inner middle*), and inner-population B (*inner bottom*)

the populations (here at level 0.25, 0.5, and 0.75 for population B) and enumerating the IIP for other population. In all cases (IIP different from 1) the system is able to stabilise at a particular shared lexicon. Interestingly, the number of iterations needed to reach coherence of 0.99 seems to be roughly identical for all analysed cases. This is due to the fact that the assumed IIP values for population B, all fall into the middle area, and the enumeration of IIP for population A seems to be irrelevant. As opposed to previous situations there seems to be a particular correlation between different IIP and the minimum coherence rate, both inner- and cross-population wise. In particular, increasing value of the IIP increases the minimum coherence rate, but only in cross-population and inner-population A wise. This behaviour is due to the fact that there is a significant disproportion of communication tendencies between two groups. In practice, population A at high values of IIP is involved in significantly more interactions, as compared to population B. Moreover, increasing the IIP value for the fixed population B seems to negatively affect the minimum coherence rate within population A, i.e., lowering its value.

We note that in all of the studied cases (also those not presented in this paper) we managed to observe a perfect alignment of lexicons utilised within inner populations towards a shared and common lexicon utilised within the entire population of interacting agents. Despite several differences in the particular behaviour of the process we can note that the presented approach, based on the language game model, allows the process of cross-population semiosis to conclude successfully.

Finally, we can focus on studying the difference between the original lexicons (as fixed at the beginning of experimentation) and established lexicons (as developed of the alignment process). In particular, we focus on two aspects of this difference: we analyse whether the resultant lexicon is either a new lexicon, or is it one of the original (A or B) (See Fig. 6—top), and we analyse the similarity between resultant lexicon and each of the original lexicons (See Fig. 6—A middle,

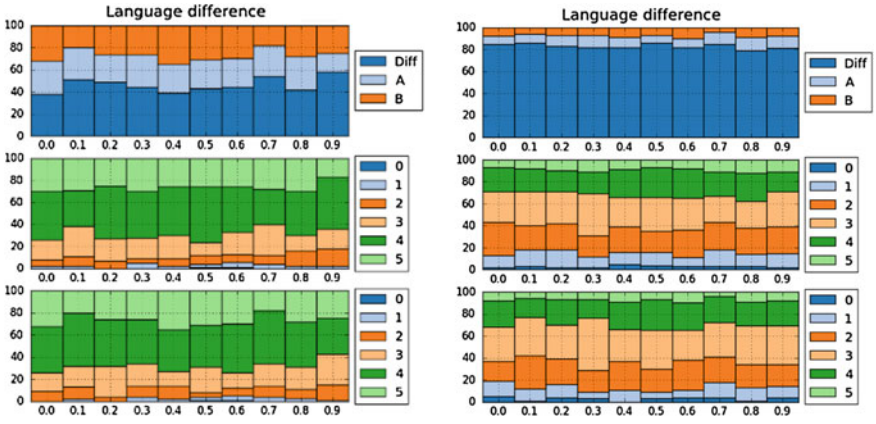

Fig. 6 Lexicon difference; small (*left*) and large (*right*) original difference; cross-population (*top*), inner-population A (*middle*), and inner-population B (*bottom*)

B bottom). Unfortunately, we can observe the tendency to formulate a novel lexicon rather than agreeing on one of the existing ones, i.e., in major of the realisations the resultant lexicon differed from the original ones. This tendency is visible (50 %) in case of small lexicon original differences, and is dominant (80 %) in case of large lexicon original differences. In the former case the internal similarity between the original lexicon and the resultant one is rather high, i.e., in 80 % of realisations the resultant lexicon differed only on a single word-object pairing (similarity above 4) and only in less than 10 % the resultant lexicon differed on more than 3 word-object pairings. Whereas in the latter case, the internal similarity is only slight, i.e., only in 20 % realisations the resultant lexicon differed on 2 or less word-object pairings and in more than 50 % realisations it differed on more than 3 word-object pairings.

As the LGM approach is suitable for the individual and population semiosis scenario, where the main focus of the linguistic development is on an individual agent, it seems not an efficient solution for the cross-population scenario. In particular, the three layer structure of the model (individual level, inner-population layer and population level) requires a more explicit representation in the alignment process mechanism and in the internal organisation of the individual agent. For instance, allowing the agent to maintain two lexicons (bilingualism), allowing to identify the population origin of the interlocutor, and allowing to keep track of other agent's lexicons, seem to be more than beneficial in the case of cross-population semiosis scenario.

4 Conclusions

In this paper we managed to pinpoint the approach and the general model of cross-population semiosis. Further, utilising the language game model we studied the behaviour of the alignment process in three general situations. In particular, using an agent based simulation framework we managed to experimentally verify the appropriateness of classical language game model mechanism in case of cross-population semiosis.

Despite the inherent simplification the obtained results provide a general overview of behaviour of the language game model in a complex, but reasonable and common-sense, cross-population scenario. Moreover, they provide a good foundation for future studies on the dynamics of language alignment process in cross-population setting, and allow to pursue the goal to define the needed settings for resultant formulation of consistent and common substance of symbols, i.e. conventionalised symbols. Consequently, we definitely extend the area of possible application of the mechanism and fill in the current gap in the research, as the cross-population semiosis settings were neglected in the literature.

Unfortunately, the obtained results show that a classical approach of LGM is not really suitable to cover the case of cross-population semiosis. Although, in all studied cases the system managed to reach a coherent formulation of lexicons, we notice that the resultant structures differed highly from their original counterparts.

Secondly, during the alignment phase the system loses the ability to maintain coherent inner population lexicons. Moreover, even in a simple case of only a small difference between the inner population the amount of iterations that is required to reach a coherent state is significant.

The aforementioned weaknesses of the LGM approach are due to the fact that the interaction in LGM is realised solely on an agent level (individual). As such the agent even has no direct means to infer the origin of its interlocutor. We note that it is necessary to introduce a more complex model. A model that would take into consideration agent's source population (agent's identity), store reflections of other agent's lexicons (agent's reflections), and would allow an agent to maintain multiple lexicons (multilingualism). However, such a modification would certainly require a major reorganisation of the entire model.

Acknowledgements The publication is financed from the statutory activities of the Faculty of Computer Science and Management of Wroclaw University of Technology.

References

1. Cook, D., Das, S.: How smart are our environments? An updated look at the state of the art. Pervasive and Mobile Computing **3**(2), 53–73 (2007)
2. Hong, X., Nugent, C., Mulvenna, M., McClean, S., Scotney, B., Devlin, S.: Evidential fusion of sensor data for activity recognition in smart homes. Pervasive Mob. Comput. **5**(3), 236–252 (2009)
3. Wark, T., Swain, D., Crossman, C., Valencia, P., Bishop-Hurley, G., Handcock, R.: Sensor and Actuator Networks: Protecting Environmentally Sensitive Areas. IEEE Pervasive Comput. **8**(1), 30–36 (2009)
4. Mirolli, M., Nolfi, S.: Evolving communication in embodied agents: theory, methods, and evaluation. In: Evolution of Communication and Language in Embodied Agents, pp. 105–121 (2010)
5. Rekleitis, I.: Distributed coverage with multi-robot system. In: Proceedings ICRA'06, p. 2423–2429 (2006)
6. Steels, L.: Language as a complex adaptive system. In: Parallel Problem Solving from Nature PPSN VI, pp. 17–26. Springer, Berlin (2000)
7. Steels, L.: Modeling the formation of language in embodied agents: methods and open challenges. In: Evolution of Communication and Language in Embodied Agents, pp. 223–233 (2010)
8. de Saussure, F.: Course in general linguistics La Salle, IL.: Open Court (1983)
9. Lorkiewicz, W., Katarzyniak, R.: Issues on aligning the meaning of symbols in multiagent systems. In: New Challenges in Computational Collective Intelligence, Studies in Computational Intelligence, vol. 244, pp. 217–229. Springer, Berlin (2009)
10. Lorkiewicz, W., Katarzyniak, R., Kowalczyk, R.: Individual semiosis in multi-agent systems. In: Transactions on Computational Collective Intelligence VII (Lecture Notes in Computer Science, vol. 7270) pp. 164–197 (2010)
11. Lorkiewicz, W., Katarzyniak, R.: Multi-stage and multi-participant interaction patterns in multi-agent naming game. Comput. Methods Sci. Technol. **20**(2), 59–80 (2014)
12. Cangelosi, A.: The grounding and sharing of symbols. In: Cognition Distributed: How Cognitive Technology Extends Our Minds, p. 83 (2008)
13. Cangelosi, A., Parisi, D.: Simulating the evolution of language. Springer, NY (2002)

14. Vogt, P., Coumans, H.: Investigating social interaction strategies for bootstrapping lexicon development. J. Artif. Soc. Soc. Simul. **6**(1), 1 (2003)
15. DeVylder, B., Tuyls, K.: Towards a common lexicon in the naming game: the dynamics of synonymy reduction. In: Workshop on Semiotic Dynamics of Language Games (2005)
16. DeBeule, J., DeVylder, B., Belpaeme, T.: A cross-situational learning algorithm for damping homonymy in the guessing game. In: Proceedings of ALIFE X. MIT Press, Cambridge (2006)
17. Lorkiewicz, W., Kowalczyk, R., Katarzyniak, R., Vo Q.B.: On topic selection strategies in multi-agent naming game. In: Proceedings of AAMAS 2011, vol. 2, pp. 499–506 (2011)

Author Index

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0