International Journal on Advances in Networks and Services

IARIA

2015 vol. 8 nr. 3&4

The *International Journal on Advances in Networks and Services* is published by IARIA.
ISSN: 1942-2644
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Networks and Services
Volume 8, Number 3 & 4, 2015

Mahmoud Brahimi, University of Msila, Algeria
Marco Bruti, Telecom Italia Sparkle S.p.A., Italy
Dumitru Burdescu, University of Craiova, Romania
Diletta Romana Cacciagrano, University of Camerino, Italy
Maria-Dolores Cano, Universidad Politécnica de Cartagena, Spain
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain
Eduardo Cerqueira, Federal University of Para, Brazil
Bruno Chatras, Orange Labs, France
Marc Cheboldaeff, T-Systems International GmbH, Germany
Kong Cheng, Vencore Labs, USA
Dickson Chiu, Dickson Computer Systems, Hong Kong
Andrzej Chydzinski, Silesian University of Technology, Poland
Hugo Coll Ferri, Polytechnic University of Valencia, Spain
Noelia Correia, University of the Algarve, Portugal
Noël Crespi, Institut Telecom, Telecom SudParis, France
Paulo da Fonseca Pinto, Universidade Nova de Lisboa, Portugal
Philip Davies, Bournemouth and Poole College / Bournemouth University, UK
Carlton Davis, École Polytechnique de Montréal, Canada
Claudio de Castro Monteiro, Federal Institute of Education, Science and Technology of Tocantins, Brazil
João Henrique de Souza Pereira, University of São Paulo, Brazil
Javier Del Ser, Tecnalia Research & Innovation, Spain
Behnam Dezfouli, Universiti Teknologi Malaysia (UTM), Malaysia
Daniela Dragomirescu, LAAS-CNRS, University of Toulouse, France
Jean-Michel Dricot, Université Libre de Bruxelles, Belgium
Wan Du, Nanyang Technological University (NTU), Singapore
Matthias Ehmann, Universität Bayreuth, Germany
Wael M El-Medany, University Of Bahrain, Bahrain
Imad H. Elhajj, American University of Beirut, Lebanon
Gledson Elias, Federal University of Paraíba, Brazil
Joshua Ellul, University of Malta, Malta
Rainer Falk, Siemens AG - Corporate Technology, Germany
Károly Farkas, Budapest University of Technology and Economics, Hungary
Huei-Wen Ferng, National Taiwan University of Science and Technology - Taipei, Taiwan
Gianluigi Ferrari, University of Parma, Italy
Mário F. S. Ferreira, University of Aveiro, Portugal
Bruno Filipe Marques, Polytechnic Institute of Viseu, Portugal
Ulrich Flegel, HFT Stuttgart, Germany
Juan J. Flores, Universidad Michoacana, Mexico
Ingo Friese, Deutsche Telekom AG - Berlin, Germany
Sebastian Fudickar, University of Potsdam, Germany
Stefania Galizia, Innova S.p.A., Italy
Ivan Ganchev, University of Limerick, Ireland
Miguel Garcia, Universitat Politecnica de Valencia, Spain
Emiliano Garcia-Palacios, Queens University Belfast, UK
Marc Gilg, University of Haute-Alsace, France
Debasis Giri, Haldia Institute of Technology, India

Markus Goldstein, Kyushu University, Japan
Luis Gomes, Universidade Nova Lisboa, Portugal
Anahita Gouya, Solution Architect, France
Mohamed Graiet, Institut Supérieur d'Informatique et de Mathématique de Monastir, Tunisie
Christos Grecos, University of West of Scotland, UK
Vic Grout, Glyndwr University, UK
Yi Gu, Middle Tennessee State University, USA
Angela Guercio, Kent State University, USA
Xiang Gui, Massey University, New Zealand
Mina S. Guirguis, Texas State University - San Marcos, USA
Tibor Gyires, School of Information Technology, Illinois State University, USA
Keijo Haataja, University of Eastern Finland, Finland
Gerhard Hancke, Royal Holloway / University of London, UK
R. Hariprakash, Arulmigu Meenakshi Amman College of Engineering, Chennai, India
Go Hasegawa, Osaka University, Japan
Eva Hladká, CESNET & Masaryk University, Czech Republic
Hans-Joachim Hof, Munich University of Applied Sciences, Germany
Razib Iqbal, Amdocs, Canada
Abhaya Induruwa, Canterbury Christ Church University, UK
Muhammad Ismail, University of Waterloo, Canada
Vasanth Iyer, Florida International University, Miami, USA
Peter Janacik, Heinz Nixdorf Institute, University of Paderborn, Germany
Imad Jawhar, United Arab Emirates University, UAE
Aravind Kailas, University of North Carolina at Charlotte, USA
Mohamed Abd rabou Ahmed Kalil, Ilmenau University of Technology, Germany
Kyoung-Don Kang, State University of New York at Binghamton, USA
Sarfraz Khokhar, Cisco Systems Inc., USA
Vitaly Klyuev, University of Aizu, Japan
Jarkko Kneckt, Nokia Research Center, Finland
Dan Komosny, Brno University of Technology, Czech Republic
Ilker Korkmaz, Izmir University of Economics, Turkey
Tomas Koutny, University of West Bohemia, Czech Republic
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lars Krueger, T-Systems International GmbH, Germany
Kae Hsiang Kwong, MIMOS Berhad, Malaysia
KP Lam, University of Keele, UK
Birger Lantow, University of Rostock, Germany
Hadi Larijani, Glasgow Caledonian Univ., UK
Annett Laube-Rosenpflanzer, Bern University of Applied Sciences, Switzerland
Gyu Myoung Lee, Institut Telecom, Telecom SudParis, France
Shiguo Lian, Orange Labs Beijing, China
Chiu-Kuo Liang, Chung Hua University, Hsinchu, Taiwan
Wei-Ming Lin, University of Texas at San Antonio, USA
David Lizcano, Universidad a Distancia de Madrid, Spain
Chengnian Long, Shanghai Jiao Tong University, China
Jonathan Loo, Middlesex University, UK

Rajarshi Sanyal, Belgacom International Carrier Services, Belgium
Mohamad Sayed Hassan, Orange Labs, France
Thomas C. Schmidt, HAW Hamburg, Germany
Hans Scholten, Pervasive Systems / University of Twente, The Netherlands
Véronique Sebastien, University of Reunion Island, France
Jean-Pierre Seifert, Technische Universität Berlin & Telekom Innovation Laboratories, Germany
Sandra Sendra Compte, Polytechnic University of Valencia, Spain
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Roman Y. Shtykh, Rakuten, Inc., Japan
Salman Ijaz Institute of Systems and Robotics, University of Algarve, Portugal
Adão Silva, University of Aveiro / Institute of Telecommunications, Portugal
Florian Skopik, AIT Austrian Institute of Technology, Austria
Karel Slavicek, Masaryk University, Czech Republic
Vahid Solouk, Urmia University of Technology, Iran
Peter Soreanu, ORT Braude College, Israel
Pedro Sousa, University of Minho, Portugal
Vladimir Stantchev, SRH University Berlin, Germany
Radu Stoleru, Texas A&M University - College Station, USA
Lars Strand, Nofas, Norway
Stefan Strauβ, Austrian Academy of Sciences, Austria
Álvaro Suárez Sarmiento, University of Las Palmas de Gran Canaria, Spain
Masashi Sugano, School of Knowledge and Information Systems, Osaka Prefecture University, Japan
Young-Joo Suh, POSTECH (Pohang University of Science and Technology), Korea
Junzhao Sun, University of Oulu, Finland
David R. Surma, Indiana University South Bend, USA
Yongning Tang, School of Information Technology, Illinois State University, USA
Yoshiaki Taniguchi, Kindai University, Japan
Anel Tanovic, BH Telecom d.d. Sarajevo, Bosnia and Herzegovina
Olivier Terzo, Istituto Superiore Mario Boella - Torino, Italy
Tzu-Chieh Tsai, National Chengchi University, Taiwan
Samyr Vale, Federal University of Maranhão - UFMA, Brazil
Dario Vieira, EFREI, France
Lukas Vojtech, Czech Technical University in Prague, Czech Republic
Michael von Riegen, University of Hamburg, Germany
You-Chiun Wang, National Sun Yat-Sen University, Taiwan
Gary R. Weckman, Ohio University, USA
Chih-Yu Wen, National Chung Hsing University, Taichung, Taiwan
Michelle Wetterwald, HeNetBot, France
Feng Xia, Dalian University of Technology, China
Kaiping Xue, USTC - Hefei, China
Mark Yampolskiy, Vanderbilt University, USA
Dongfang Yang, National Research Council, Canada
Qimin Yang, Harvey Mudd College, USA
Beytullah Yildiz, TOBB Economics and Technology University, Turkey
Anastasiya Yurchyshyna, University of Geneva, Switzerland
Sergey Y. Yurish, IFSA, Spain

Jelena Zdravkovic, Stockholm University, Sweden
Yuanyuan Zeng, Wuhan University, China
Weiliang Zhao, Macquarie University, Australia
Wenbing Zhao, Cleveland State University, USA
Zibin Zheng, The Chinese University of Hong Kong, China
Yongxin Zhu, Shanghai Jiao Tong University, China
Zuqing Zhu, University of Science and Technology of China, China
Martin Zimmermann, University of Applied Sciences Offenburg, Germany

International Journal on Advances in Networks and Services
Volume 8, Numbers 3 & 4, 2015

CONTENTS

Eduardo Magana, Public University of Navarre, Spain
Daniel Morato, Public University of Navarre, Spain

pages: 192 - 202
Deriving Robust Distributed Business Processes with Automated Transformations of Fallible Component Processes
Lei Wang, University of Twente, the Netherlands
Luis Pires, University of Twente, the Netherlands
Marten van Sinderen, University of Twente, the Netherlands
Andreas Wombacher, Achmea, the Netherlands
Chi-Hung Chi, CSIRO, Australia

pages: 203 - 214
How to Operate Container Clusters more Efficiently? Some Insights Concerning Containers, Software-Defined-Networks, and their sometimes Counterintuitive Impact on Network Performance
Nane Kratzke, Lübeck University of Applied Sciences, Germany
Peter-Christian Quint, Lübeck University of Applied Sciences, Germany

pages: 215 - 225
Distance Sensor Assistance to GPS Positioning
Yasuhiro Ikeda, Utsunomiya University, Japan
Hiroyuki Hatano, Utsunomiya University, Japan
Masahiro Fujii, Utsunomiya University, Japan
Atsushi Ito, Utsunomiya University, Japan
Yu Watanabe, Utsunomiya University, Japan
Tomoya Kitani, Shizuoka University, Japan
Toru Aoki, Shizuoka University, Japan
Hironobu Onishi, Shizuoka University, Japan

Bounded Side-based Clustering VBF Routing Protocol in Underwater Wireless Sensor Networks

Dina M. Ibrahim, Tarek E. Eltobely, Mahmoud M. Fahmy, and Elsayed A. Sallam
Computers and Control Engineering Department
Faculty of Engineering, Tanta University
Tanta, Egypt
emails: {dina.mahmoud@f-eng.tanta.edu.eg, tarekt@f-eng.tanta.edu.eg, mfn_288@hotmail.com, and sallam@f-eng.tanta.edu.eg}

Abstract— **Underwater Wireless Sensor Networks (UWSNs) have an important role in different applications, such as offshore exploration and ocean monitoring. In this paper, we improve the performance of the Clustering Vector-Based Forwarding algorithm (CVBF) by enhancing its behavior. Here, the whole network is divided into a predefined number of clusters and determining an internal bounded side with its own coordinates for each cluster. The main target of our proposed algorithm is to avoid the calculations taken for the nodes in which they never move away from its cluster. These nodes' positions are near the cluster axis and far enough from the cluster boundaries. For this reason, we separate between the nodes that can move outside the clusters and the nodes that never move away. Simulation results demonstrate that the proposed algorithm reduces the energy consumption especially in dense networks, increases the packet delivery ratio especially in sparse networks, and decreases the average end-to-end delay in both sparse and dense networks. These advantages are emphasized when the algorithm is compared with five other powerful routing algorithms: VBF, Hop-by-Hop VBF (HH-VBF), Vector-Based Void Avoidance (VBVA), Energy-Saving VBF (ES-VBF), and the CVBF routing protocols.**

Keywords-wireless networks; underwater sensor networks; multiple clusters; routing protocols; CVBF; BS-CVBF.

I. INTRODUCTION

At the end of the twentieth century, wireless sensor networks became a hot research area. At the beginning, these networks covered only terrestrial applications. However, the earth is known to be a water planet, with 70 % of its surface being covered with water (principally oceans). With the increasing role of oceans in human life, discovering all of the ocean parts became of prime importance. On one side, traditional approaches formerly used for underwater monitoring missions have several drawbacks and on the other side, these harsh environments are not feasible for human presence as unpredictable underwater activities, high water pressure, predatory fish and vast areas are major reasons for un-manned exploration. Due to these reasons, Underwater Wireless Sensor Networks (UWSNs) attract the interest of many researchers lately [1]. Over the last three decades, significant contribution has been made in the area of scientific, commercial, and military applications [2][3][4][5]. In particular, highly precise real-time continuous-monitoring systems are essential for vital operations such as off-shore oil field monitoring, pollution detection, disaster prevention, assisted navigation, mine reconnaissance, and oceanographic data collection [6]. All these significant applications call for building UWSNs. The work done by Akylidiz et al. [7] is considered as the pioneering effort towards the deployment of sensor nodes for underwater environments.

Though there exist many network protocols for terrestrial wireless sensor networks, the underwater acoustic communication channel has its unique characteristics, such as limited bandwidth capacity and high delays, which require new efficient and reliable data communication protocols [8][9][10]. Major challenges in the design of underwater wireless sensor networks are: i) the limited bandwidth; ii) the underwater channel is severely impaired, especially due to multipath and fading problems; iii) high propagation delay in underwater which is five orders of magnitude higher than in Radio Frequency (RF) terrestrial channels; iv) high energy consumption due to longer distances; v) battery power is limited and usually batteries cannot be recharged, also because solar energy cannot be exploited underwater; vi) underwater sensor nodes are prone to failures due to fouling and corrosion. All the factors mentioned above, especially limited energy, would make designing a routing protocol for UWSN an enormous challenge.

Routing is a fundamental issue for any network, and routing protocols are considered to be in charge of discovering and maintaining the routes [6]. Most of the research works concerning UWSNs have been on the issues related to the physical layer, while issues related to the network layer such as routing techniques are a relatively new area. Thus, an efficient routing algorithm is to be provided. Although underwater acoustic networks have been studied for decades, underwater networking and routing protocols are still at the infant stage of research.

A review of underwater network protocols till the year 2000 can be found in [2]. Several routing protocols have been proposed for underwater sensor networks. A good survey until year 2012 about underwater wireless sensor routing techniques is presented in [6]. Here, Ayaz et al. introduced an overview of the state of the art of routing protocols in UWSNs and thoroughly highlighted the advantages, functionalities, weaknesses and performance issues for each technique. Based on network architecture, UWSNs routing protocols are classified into: location-based, flat, and hierarchical routing protocols. Vector-Based Forwarding (VBF) protocol has been suggested in order to solve the problem of high error probability in dense networks [11]. It is a location-based

routing protocol. Here an idea of a virtual routing pipe from the source to the destination is proposed, and all the flooding data packets are carried out through this pipe. An enhanced version of VBF called Hop-by-Hop VBF (HH-VBF) has been proposed [12]. They use the same concept of virtual routing pipe as used by VBF, but instead of using a single pipe from source to destination, HH-VBF defines per hop virtual pipe for each forwarder [13]. Another extension of VBF protocol is introduced in [14] called Vector-Based Void Avoidance (VBVA) routing protocol, which extends the VBF routing protocol. It addresses the routing void problem in underwater sensor networks. VBVA assumes two mechanisms, vector-shift and back-pressure, to handle voids. In [15], an energy-aware routing algorithm, called Energy-Saving Vector Based Protocol (ES-VBF), is proposed. In this protocol, Bo et al. put forward an energy-aware routing algorithm to save network energy. It takes both residual energy and location information into consideration, which shows a promising performance in balancing network energy consumption and packet reception ratio.

In our recent research [1], we propose a Clustering Vector-Based Forwarding algorithm (CVBF) in order to improve the performance of the VBF protocol. In the proposed algorithm, the network space volume is divided into a number of clusters where one virtual sink is assigned to each cluster. Inside each cluster, all the nodes are allowed to communicate with themselves just to reach its virtual sink node, which in turn sends the packets to the main sink in the network. Simulation results demonstrate that the proposed algorithm reduces the energy consumption especially in dense networks, increases the packet delivery ratio especially in sparse networks, and decreases the average end-to-end delay in both sparse and dense networks. These advantages are emphasized when the algorithm is compared with five other powerful routing algorithms: VBF, HH-VBF, VBVA, ES-VBF, and CVBF routing protocols.

Other UWSNs routing protocols, such as Dynamic Source Routing (DSR), Focused beam Routing (FBR), Directional Flooding-Based (DFR), and Depth-Based Routing (DBR) are found in [6][13][16][17].

The remainder of this paper is organized as follows. In Section II, the behavior and performance issues of VBF, HH-VBF, VBVA, ES-VBF, and CVBF location-based routing protocols, which will be used in a comparison with our algorithm, are discussed. Section III presents the details of the proposed algorithm. In Section IV, the performance under varying the number of clusters are discussed. Then, we show the performance results of the proposed algorithm in Section V. Finally, we draw the main conclusions in Section VI.

II. LOCATION-BASED ROUTING PROTOCOLS: A REVIEW

In this section, we briefly discuss the five location-based routing protocols, which we will select to compare our algorithm with. These protocols are presented in the following:

A. Vector-Based Forwarding (VBF) Routing Protocol

VBF is a location-based routing approach for UWSNs proposed by Xie et al. [11]. In this protocol, state information of the sensor nodes is not required since only a small number of nodes are involved during packet forwarding. Data packets are forwarded along redundant and interleaved paths from the source to the sink, which helps handling the problem of packet losses and node failures. It is assumed that every node previously knows its location, and each packet carries the location of all the nodes involved including the source, forwarding nodes, and final destination. The forwarding path is specified by the routing vector from the sender to the target. As soon as a packet is received, the node computes its relative position with respect to the forwarder. Recursively, all the nodes receiving the packet compute their positions. If a node determines that it is close enough to the routing vector, it puts its own computed position in the packet and continues forwarding the packet; else, it simply discards the packet. In this way, all the packet forwarders in the sensor network form a "routing pipe", the sensor nodes in this pipe are eligible for packet forwarding, and those that are not close to the routing vector do not forward. Fig. 1 illustrates the basic idea of VBF. In this figure, node S_1 is the source, and node S_0 is the sink. The routing vector is specified by S_1S_0. Data packets are forwarded from S_1 to S_0. Forwarders along the routing vector form a routing pipe with a pre-controlled radius, W.

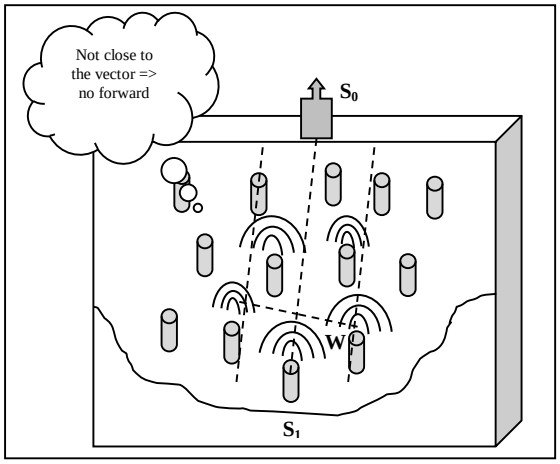


Figure 1. VBF routing protocol for UWSNs.

Additionally, a localized and distributed self-adaptation algorithm is developed to enhance the performance of VBF [11]. The self-adaptation algorithm allows each node to estimate the density in its neighborhood and forward packets adaptively. This algorithm is based on the definition of a desirableness factor, α [11]. This factor measures the suitability of a node to forward packets. Given a routing vector S_1S_0 and forwarder F, the desirableness factor of a node A is:

$$\propto = \frac{P}{W} + \frac{R - d \times \cos\theta}{R} \quad (1)$$

where P is the projection length of A onto the routing vector S_1S_0, d is the distance between node A and node F, θ is the

angle between vector FS_0 and vector FA, R is the transmission range, and W is the radius of the routing pipe.

VBF has many essential drawbacks. First, using a virtual routing pipe from source to destination can affect the routing efficiency of the network with different node densities. In some spaces, if node deployment is sparser or becomes sparse due to some node movement, then it is possible that very few or even no node will lie within that virtual pipe, which is responsible for the data forwarding; even it is possible that some paths may exist outside the pipe. Eventually, this will result in small data deliveries in sparse spaces. Second, VBF is very sensitive about the routing pipe radius threshold, and this threshold can affect the routing performance significantly; such feature may not be desirable in the real protocol developments. Furthermore, some nodes along the routing pipe are used again and again in order to forward the data packets from sources to the sink, which can exhaust their battery power.

B. HH-VBF Routing Protocol

The need to overcome two problems encountered by the VBF, i.e., small data delivery ratio in sparse networks, and sensitivity to the routing pipe's radius, the HH-VBF (hop-by-hop VBF) is proposed by Nicolaou et al. [12]. HH-VBF forms the routing pipe in a hop-by-hop method, enhancing the packet delivery ratio significantly. Although it is based on the same concept of routing vector as VBF, instead of using a single virtual pipe from the source to the sink, it defines a different virtual pipe around the per-hop vector from each forwarder to the sink. In this protocol, each node can adaptively make packet forwarding decisions based on its current location. This design can directly bring the following two benefits: First, since each node has its own routing pipe, the maximum pipe radius is the transmission range. Second, in sparse networks, HH-VBF can find a data delivery path even so the number of eligible nodes may be small, as long as there exists one in the network.

In HH-VBF, the routing virtual pipe is redefined to be a per-hop virtual pipe, instead of a unique pipe from the source to the sink [12]. When some areas of the network are not occupied with nodes, for example, there exist "voids" in the network, even a self-adaptation algorithm may not be able to route the packets. In such a case, a forwarder is unable to reach any node other than the previous hop. Although simulation results show that HH-VBF considerably produces better results for packet delivery ratio, but still it has an inherent problem of routing pipe radius threshold, which can affect its performance. Moreover, due to its hop-by-hop nature, HH-VBF is not able to add a feedback mechanism to detect and avoid voids in the network and energy efficiency is still low compared to VBF [12].

C. VBVA Routing Protocol

Xie et al. [14] introduce a Vector-Based Void Avoidance (VBVA) routing protocol, which extends the VBF routing protocol to handle the routing void problem in UWSNs. VBVA assumes two mechanisms, vector-shift and back-pressure. The vector-shift mechanism is used to route data packets along the boundary of a void. The back-pressure mechanism routes data packets backward to bypass a concave void. VBVA handles the routing void problem on demand and thus does not need to know network topology and void information in advance. Hence, it is very robust to cope with mobile voids in mobile networks. Simulation results in [14] show that VBVA can handle both concave and convex voids effectively and efficiently in mobile underwater sensor networks only when these voids are inside the forwarding pipe, while the voids outside the forwarding pipe is not solved by VBVA.

D. ES-VBF Routing Protocol

To solve the energy problem in UWSN, Bo et al. [15] put forward an energy-aware routing algorithm, called Energy-Saving Vector-Based Protocol (ES-VBF). The main purpose of this routing protocol is saving energy. ES-VBF takes both residual energy and localization-based information into consideration while calculating the desirableness factor as in (2), which allows nodes to weigh the benefit for forwarding packets. The ES-VBF algorithm modifies the calculation of the desirableness factor of (1) for VBF protocol to be calculated if the node residual energy is smaller than 60% of initial energy as:

$$\propto = 0.5 \times \left(1 - \frac{energy}{initialenergy}\right) + \left(\frac{P}{W}\right) + \left(\frac{R - d \times \cos\theta}{R}\right) \quad (2)$$

where *energy* is the residual energy of nodes and *initialenergy* is the initial energy of nodes. By simulation results in [15], it is shown that the performance is promising in balancing network energy consumption and packet reception ratio. This means that the ES-VBF protocol saves energy in an efficient manner. At the same time, there is a small falling in packet reception ratio, which needs further research aiming at finding a better solution not only reducing energy consumption but also achieving high packet reception ratio.

E. Clustering VBF (CVBF)Routing Protocol

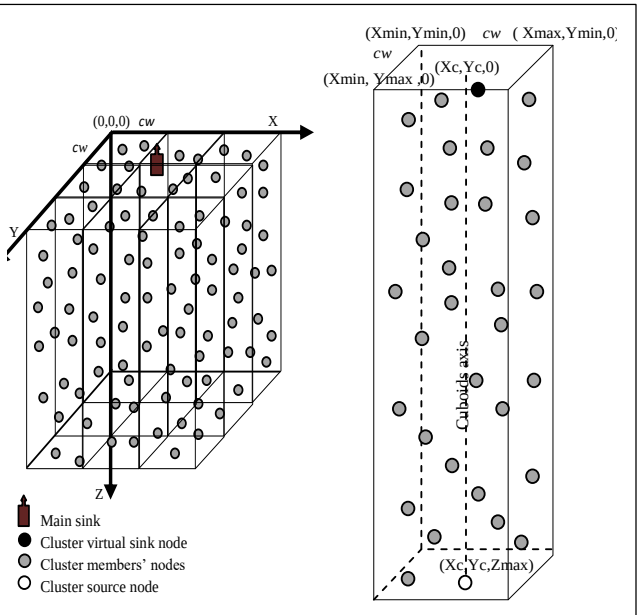
In this algorithm (our recent research) the authors propose an algorithm for UWSNs called CVBF algorithm. The objective of the CVBF routing algorithm is to reduce energy consumption, increase the packet delivery ratio, and decrease the average end-to-end delay.

In [1], the whole network is divided into a predefined number of clusters. All sensor nodes are assigned to the clusters on the basis of their geographic location, and then one node at the top of each cluster is selected as a *virtual sink* for that cluster. The rest of nodes in each cluster transmit the data packets to their respective cluster virtual sink. The routing inside each cluster follows the VBF routing protocol discussed in Section II. This implies that the concept of using one virtual routing pipe for all network nodes in VBF is replaced by defining one virtual routing pipe for each cluster to forward the packets from any node in the cluster to its virtual sink in that cluster. We assume that the routing pipe radius is equal to the transmission range of a node. Each

intermediate node in any cluster selects the next hop to a node inside its cluster. In this way, the network will have many virtual routing pipes, one pipe per cluster, which guarantees forwarding the packets in the upward direction instead of forwarding the packets widely across the network nodes in the VBF algorithm. It is well expected that this will decrease the average end-to-end delay node and reduce the number of hops to reach the virtual sink node, which will enhance the network performance. In addition, CVBF avoid voids in the network because each node belongs to a specific cluster.

Also, if a small number of nodes are available in the neighborhood, CVBF can still find a data delivery path. After receiving the data packets from cluster sensor nodes, cluster virtual sinks perform an aggregation function on the received data, and transmit them towards the main sink node using single-hop routing. Cluster virtual sink nodes are responsible for coordinating their cluster members and communicating with the main sink node. The proposed algorithm consists of 3 steps: Step1, which called *clustering the nodes*, step 2 is *selecting the cluster virtual sink*, and step 3, which called *calculating the cluster's maintenance time*.

Step 1 is responsible for dividing the network into groups of nodes according to their geographic location producing non-overlapping clusters excluding the main network sink, which is allocated on the water surface. The following values are given: the network space X×Y×Z, node transmission range, routing pipe width, and the node speed. We divide the network space into equal space volumes; in the form of cuboids [1]. The division is based on the values of X and Y coordinates, and the cluster width, *cw*, as shown in Fig. 2 (a).



(a) Network area with 9 clusters (b) One cluster and its virtual sink

Figure 2. A CVBF network area: (a) Network area with 9 clusters, (b) One cluster and its virtual sink

Choosing the best number of clusters is proposed as:

$$N = \frac{X \times Y}{(cw)^2} \quad (3)$$

where $X \times Y$ is the total surface area of the network and $(cw)^2$ is the area of the cluster surface. The cluster width is thus calculated as:

$$cw = \sqrt{\left(\frac{X \times Y}{N}\right)} \quad (4)$$

It is given that the surface area is square; therefore, we choose N as a number raised to the power of two: 2^2, 3^2, 4^2, or 5^2. Here, we choose N that gives the value of *cw* as near as possible to $\sqrt{2}$R in order to make sure that the virtual pipe of the cluster includes all the nodes inside that cluster.

Step 2 selects the cluster virtual sink for each cluster (cuboid). Each cluster has a space volume of cw×cw×Z, we choose the nearest node to the main sink to be a cluster virtual sink. As shown in Fig. 2 (b), the surface corner coordinates of the cluster are: (Xmin, Ymin, 0), (Xmax, Ymin, 0), (Xmin, Ymax, 0), and (Xmax, Ymax, 0).

All other nodes can send data to their corresponding virtual sink following the mechanism of VBF and depending on the value of its desirableness factor α. If more than one node has the same depth position, we choose the nearest node to the cuboid axis, in which its surface point coordinates is the point (Xc, Yc, 0). The source node of the cluster is fixed at the position (Xc, Yc, Zmax).

Step 3 takes into consideration the node mobility that affects network topology and performance, thus necessitating a cluster maintenance algorithm. For a correct network operation, the maintenance algorithm should be executed simultaneously in all clusters. In this step, we propose a suitable periodical time, which we call *maintenance time*, *Tm*. This time is enough to move a node from its cluster to another cluster according to speed and maximum distance of the node. Each node in the cluster checks its belonging to that cluster after the periodical time *Tm*. If a node belonging to a cluster moves away from that cluster, it naturally has two choices. The first choice is to enter another neighboring cluster, and so we transfer this node from the old cluster to the new cluster. The second choice is that it exits from all the network space, and so we leave this node in the old cluster.

To calculate *Tm*, we divide the known maximum distance of a node movement, *dmax*, by the current speed of the node, *S*:

$$Tm = \frac{dmax}{S} \quad (5)$$

In other words, all the nodes with positions near the cluster boundaries are prone to exit from their own cluster and enter to other clusters. One of the CVBF algorithm drawbacks is the calculations of *Tm* for all the nodes that take much time

while there is a wasted time calculated with the nodes resides on the middle of each cluster, which never have a chance to move away from the cluster.

III. BOUNDED SIDE-BASED CLUSTERING VBF: THE PROPOSED ALGORITHM

In this section, we propose an algorithm for the routing protocols in UWSNs, which we call a Bounded Side-based Clustering VBF (BS-CVBF), which is an extended version of our recent paper in [1]. The main objective of this algorithm is to enhance the performance of the clustering VBF. This enhancement includes the energy consumption, the packet delivery ratio (PDR), the average end-to-end delay, and the CPU utilization. These four metrics are emphasized through comparison with five location-based routing protocols; VBF, HH-VBF, VBVA, ES-VBF, and CVBF protocols.

According to our algorithm, the whole network is divided into a predefined number of clusters follows step 1 in the CVBF routing algorithm discussed in Section II. In this way, we add another substep to step 1, which is responsible for determining internal bounded side with its own coordinates for each cluster, as shown in Fig. 3. The main target of our proposed algorithm is to avoid the calculations taken for the nodes in which they never move away from its cluster. These nodes' positions are near the cluster axis and far enough from the cluster boundaries. For this reason, we separate between the nodes that can move outside the clusters and the nodes that never move away.

This action is controlled by dividing the volume area of each cluster, cuboid, into two volume areas: The first volume area, which we call the inner cuboid, bounded by the (Xsmin, Ysmin, 0), (Xsmax, Ysmin, 0), (Xsmin, Ysmax, 0), and (Xsmax, Ysmax, 0). The second volume area is the rest of the cluster volume area, which represented by a gray color in Fig. 4.

The proposed algorithm is stated in the following steps:

Step1: Clustering the Nodes and Creating the Bounded Side

In addition to step 1 in the CVBF [1] we divide each cluster into two nested cuboids, as presented in Fig. 4. The inner cuboid contains the number of nodes that never move away from the cluster at any time during the network life time. The reason is our choice to the inner cuboid boundaries is calculated based on the maximum distance of a node movement where:

$$Xsmin = Xmin + dmax \quad (6)$$

$$Xsmax = Xmax - dmax \quad (7)$$

$$Ysmin = Ymin + dmax \quad (8)$$

$$Ysmax = Ymax - dmax \quad (9)$$

Step2: Selecting the Cluster Virtual Sink

For each cluster that has a space volume cw×cw×Z, we choose the nearest node to the main sink to be a cluster virtual sink. Like the CVBF algorithm. As shown in Fig. 2 (b), the surface corner coordinates of the cluster are: (Xmin, Ymin, 0), (Xmax, Ymin, 0), (Xmin, Ymax, 0), and (Xmax, Ymax, 0).

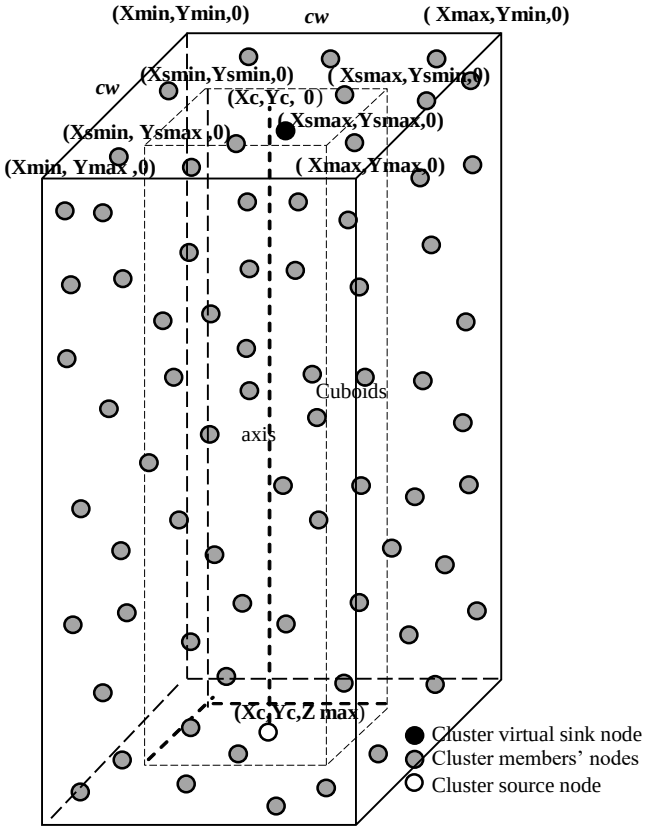


Figure 3. One cluster with the inner bounded side.

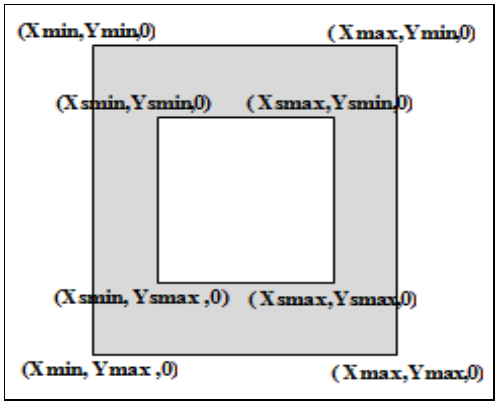


Figure 4. A horizontal section of a cluster including the inner bounded side.

All the other nodes can send data to their corresponding virtual sink following the mechanism of VBF and depending on the value of its desirableness factor α. If more than one node has the same depth position, we choose the nearest node to the cuboid axis, in which its surface point coordinates is the point (Xc, Yc, 0). The source node of the cluster is fixed at the position (Xc, Yc, Zmax).

Step3: Calculating the Cluster's Maintenance Time with respcet to the bounded side

This step takes into consideration the node mobility that affects network topology and performance, thus necessitating a cluster maintenance algorithm. For a correct network operation, the maintenance algorithm should be executed simultaneously in all clusters. In this step, we propose a suitable periodical time which we call *maintenance time, Tm.* This time is enough to move a node from its cluster to another cluster according to speed and maximum distance of the node. All the nodes inside the bounded side, represented by the gray color in Fig. 4 checks its belonging to that cluster after the periodical time *Tm*. If a node belonging to a cluster moves away from that cluster, it naturally has two choices. The first choice is to enter another neighboring cluster, and so we transfer this node from the old cluster to the new cluster. The second choice is that it exits from all the network space, and so we leave this node in the old cluster.

The proposed algorithm is summarized in the Pseudocode of Fig. 5.

Pseudocode

Step 1: Clustering the nodes and creating the bounded side

1.1. All the substeps in the CVBF algorithm

1.2. Determining the coordinates of the inner cuboid as:

(a) $Xsmin = Xmin + dmax$

(b) $Xsmax = Xmax - dmax$

(c) $Ysmin = Ymin + dmax$

(d) $Ysmax = Ymax - dmax$

Step 2: Selecting the cluster virtual sink

1. All the substeps in the CVBF algorithm

Step 3: Calculating the cluster's maintenance time with respcet to the bounded side

1. Any node is belonging to the inner cuboid (has Xis and Yis coordinates)
   if
   Ximin>=Xis>=Xsmin AND
   Yimin>=Yis>=Yimax
   OR
   Xsmax>=Xis>=Ximax AND
   Yimin>=Yis>=Yimin
   OR
   Xsmin>=Xis>=Xsmax AND
   Yimin>=Yis>=Ysmax
   OR
   Xsmin>=Xis>=Xsmax AND
   Ysmax>=Yis>=Yimax
   Then
   This node may move away from it cluster
   Else
   If a node exits from the cluster
   Then
   (a)Given the node speed, S, and the maximum distance of any node, *dmax*,
   (b) Calculate Tm=*dmax*/S
   (c) For J=0 to Simulation time with step Tm
   For each node in the cluster i and has coordinates (Xi,Yi,Z)
   If (Xsmin>=Xi>=Xsmax and Ysmin>=Yi>=Ysmax)
   Then
   This node is still in the cluster
   Else
   Remove this node from cluster i and enter it to the suitable neighboring cluster
2. All the nodes in cluster i forward the packets to its virtual sink following the mechanism of VBF routing algorithm
3. All the virtual sinks forward the packets to the main sink

Figure 5. Pseudocode of the proposed routing protocol BS-CVBF.

IV. PERFORMANCE EVALUATION

Performance is quantified through measures of energy consumption, packet delivery ratio, average end-to-end delay, and the CPU utilization percentage [15]. The energy consumption is the total energy consumed by the sensor network nodes. The packet delivery ratio is the rate of the number of packets successfully received by the sink to the number of packets generated by the source. The average delay is the average end-to-end delay for each packet received by the sink.

Simulation is performed by the underwater package Aqua-Sim of ns-2 [18][19] version ns2.35. In all our simulations, we set the parameters similar to UWM1000 LinkQuest Underwater Acoustic Modem [20]. All the network parameters are described in Table I.

TABLE I. NETWORK PARAMETERS

Parameter	***Value***
Network Space Volume	600m × 600m ×600m
Data packet size	76 bytes
Bt rate	10 kbps
Sending Energy	0.6 J
Receiving Energy	0.3 J
Idle Mode Energy	0.01 J
Node Speed	2m/s – 5m/s
Number of clusters	9 clusters
Total simulation time	1000 S
Number of the simulator running	30

The simulation results are plotted in Figs. 6, 7, and 8. Fig. 6 depicts the total energy consumption when the number of sensor nodes varies. The energy consumption increases with the number of nodes since more nodes are involved in packet forwarding. On the other hand, this figure shows that the energy consumption for the proposed algorithm is less than that in VBF and HH-VBF routing protocol only on dense networks, when the number of nodes is greater than 300 nodes, indicating that the CVBF and the BS-CVBF algorithm can save more energy with high node density.

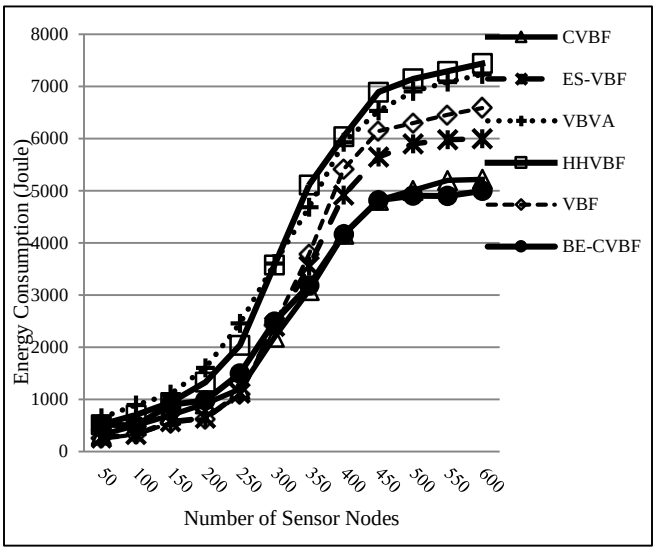


Figure 6. BS-CVBF energy consumption *vs.* number of sensor nodes.

Figure 7 shows the packet delivery ratio with the number of sensor nodes. It is seen that the packet delivery ratio increases with the increase of the number of nodes. When more than 200 nodes are deployed in the space, the packet delivery ratio remains above 90% for both ES-VBF routing protocol and BS-CVBF algorithm. Figure 7 shows that our algorithm gives better results in packet delivery ratio than VBF, HHVBF, VBVA, ES-VBF, and CVBF protocols.

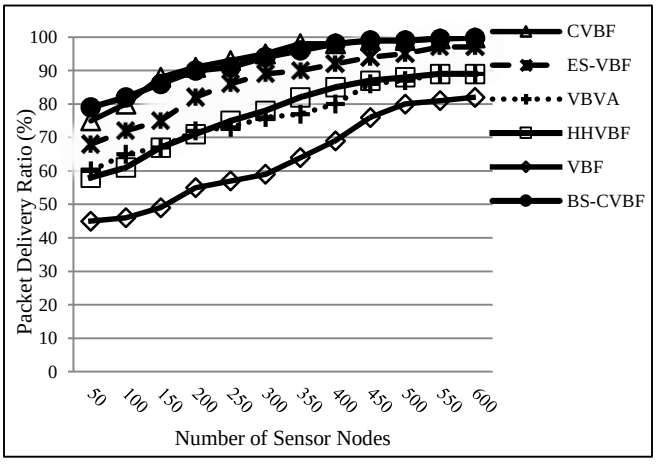


Figure 7. BS-CVBF packet delivery ratio *vs.* number of sensor nodes.

Figure 8 describes the average end-to-end delay with the number of sensor nodes. It is seen that the average end-to-end delay decreases with the increase of node density in the network. When the number of sensor nodes increases, the paths from the source to the sink are closer to the optimal path (α=0); therefore, the average end-to-end delay decreases. While Fig. 9 shows the CPU utilization with respect to the number of node.

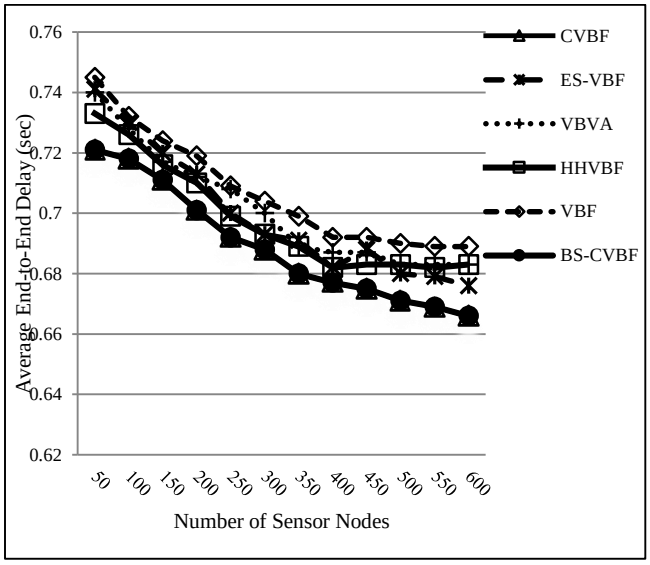


Figure 8. BS-CVBF average end-to-end delay *vs.* number of sensor nodes.

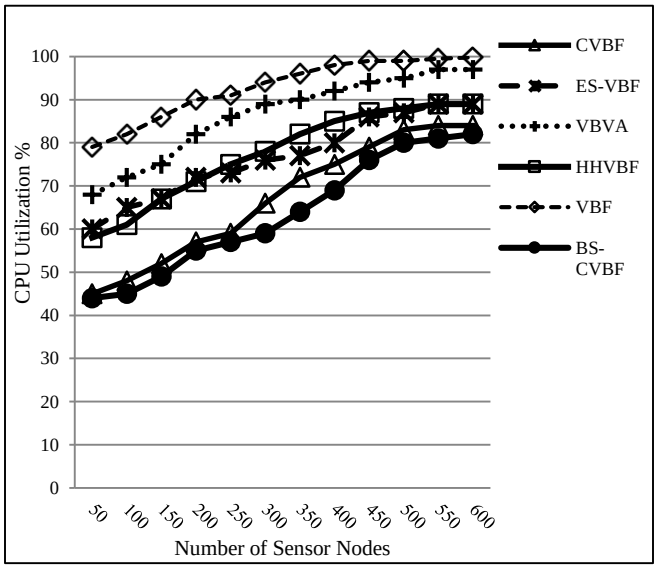


Figure 9. BS-CVBF CPU utilization *vs.* number of sensor nodes.

We evaluate the performance of BS-CVBF under various network scenarios. The simulation results show that CVBF significantly exhibits a better performance than VBF, HH-VBF, VBVA, and ES-VBF protocols since it has: lower energy consumption, higher packet delivery ratio, and lower average end-to-end delay.

Calculating the cluster width *cw* depends on two parameters: the surface area of the network X×Y and choosing the number of clusters *N*. We choose a value of *N* for which the cluster width is nearest to the value of $\sqrt{2}$R. We conclude this after examining different values of *N*. This is because each node can transmit the data packets only to the neighbors allocated in its transmission range.

V. PERFORMANCE UNDER VARYING NUMBER OF CLUSTERS

In this section, we vary the number of clusters, N, from 2^0 to 6^2 (N=1, 4, 9, 16, 25, and 36), keeping the network surface area is 600×600 m2 and R is 100m. We need to prove that the optimal value of N is the value which makes cw as near as possible to $\sqrt{2}$R. Table II shows the different values of cluster width based on equation (4) and the number of clusters.

TABLE II. CLUSTER WITH VALUES UNDER VARYING NUMBER OF CLUSTERS

Number of Clusters (*N*)	**Cluster Width (*cw*)**	**$\sqrt{2}$R (R=100)**
1	600	141.42
4	300	
9	200	
16	**150**	
25	120	
36	100	

Table II shows that the nearest value of the cluster width in this experiment is 150; this means that the preferred number of the clusters is 16 clusters. Figures 10, 11, and 12 show our simulation results in the different values of *N*, 1, 4, 9, 16, 25 and 36. Our simulation results give better performance when *N* is 16 because *cw*=150 m is the nearest value to $\sqrt{2}$R (141.42 m). It gives lower energy consumption, higher packet delivery ratio, and lower average end-to-end delay.

Figure 10 depicts the total energy consumption of the proposed BS-CVBF algorithm as the number of clusters varies. The energy consumption increases with the number of nodes since more nodes are involved in packet forwarding. On the other hand, this figure shows that the energy consumption for the proposed algorithm when the number of clusters is 16 (*N*=16) is less than the other values of *N*. This means that dividing the network into 16 clusters gives lower energy consumption.

Figure 11 shows the packet delivery ratio with the number of sensor nodes under varying the clusters number. It is seen that the packet delivery ratio increases with the increase of the number of clusters till 16 clusters and the packet delivery ratio decreases at 25 clusters.

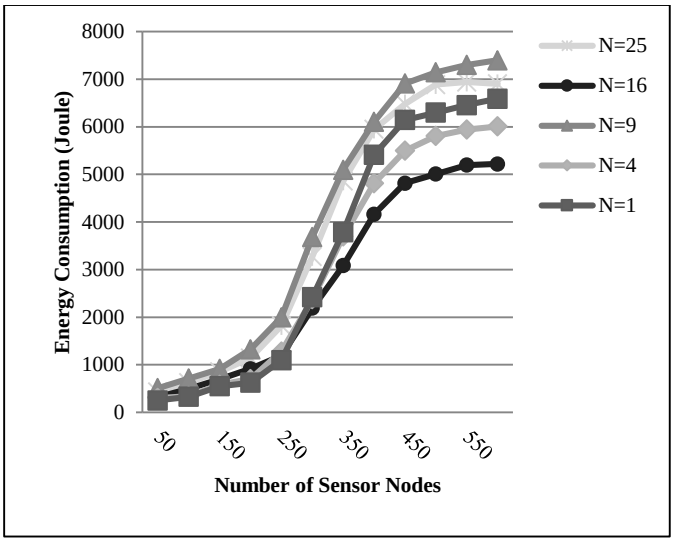


Figure 10. BS-CVBF energy consumption under varying clusters number.

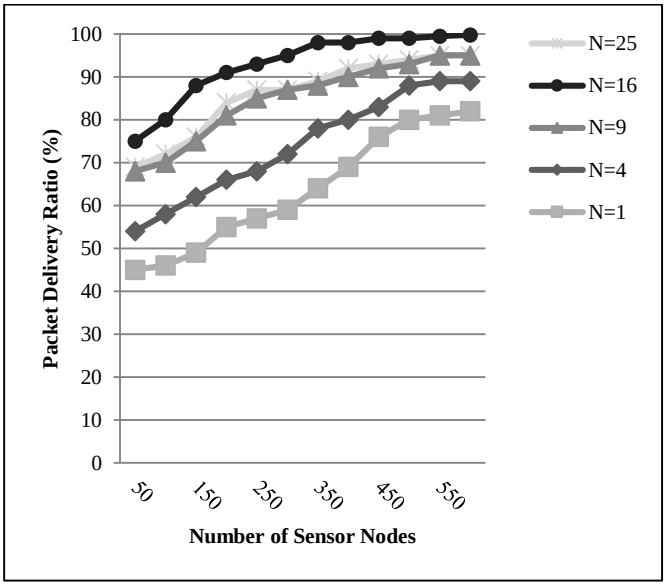


Figure 11. BS-CVBF packet delivery ratio under varying clusters number.

Figure 12 describes the average end-to-end delay with the number of sensor nodes. It is seen that the average end-to-end delay decreases with the increase of number of clusters in the network. When the number of cluster exceeds 16 clusters the average end-to-end delay increases again. Therefore, choosing *N*=16 gives better performance than that for *N*=1, 4, 9, 25.

The simulation results in Figs. 10, 11, and 12 give better performance when *N* is 16 because *cw*=150 m is the nearest value to $\sqrt{2}$R (141.42 m). It gives lower energy consumption, higher packet delivery ratio, and lower average end-to-end delay for the proposed algorithm.

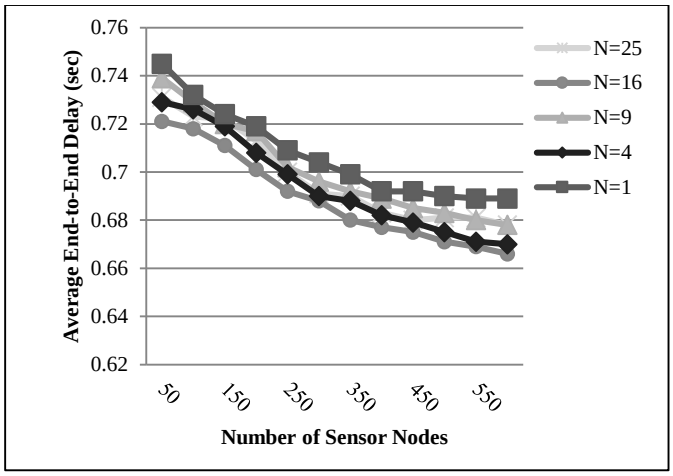


Figure 12. BS-CVBF average end-to-end delay under varying clusters number.

VI. CONCLUSIONS

In this paper, we propose a bounded side-based clustering vector-based forwarding algorithm to improve the performance of the location-based routing protocol in underwater wireless sensor networks. In the proposed algorithm, the whole network is divided into a predefined number of clusters and determining an internal bounded side with its own coordinates for each cluster. The main target of our proposed algorithm is to avoid the calculations taken for the nodes in which they never move away from its cluster. These nodes' positions are near the cluster axis and far enough from the cluster boundaries. For this reason, we separate between the nodes that can move outside the clusters and the nodes that never move away. Simulation results demonstrate that the proposed algorithm reduces the energy consumption especially in dense networks, increases the packet delivery ratio especially in sparse networks, and decreases the average end-to-end delay in both sparse and dense networks. These advantages are emphasized when the algorithm is compared with five other powerful routing algorithms: VBF, Hop-by-Hop VBF (HH-VBF), Vector-Based Void Avoidance (VBVA), Energy-Saving VBF (ES-VBF), and the CVBF routing protocols. It is interesting to note that our multiple-cluster algorithm is a good generalization to the VBF protocol. The VBF results from our algorithm by adopting the special case of single-cluster manipulation.

REFERENCES

[1] Dina M. Ibrahim, Mohmoud M. Fahmy, Tarek E. ElTobely, and ElSayed A. Sallam, "Enhancing the Vector-Based Forwarding Routing Protocol for Underwater Wireless Sensor Networks: A Clustering Approach," The Tenth International Conference on Wireless and Mobile Communications (ICWMC 2014), Seville, Spain, June, 2014, pp. 98-104.

[2] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater Acoustic Networks," IEEE Journal of Ocean Engineering, vol. 25, Jan. 2000, pp. 72-83.

[3] S. Climent, A. Sanchez, J. V. Capella, N. Meratnia, J. J. Serrano, "Underwater Acoustic Wireless Sensor Networks: Advances and Future Trends in Physical, MAC and Routing Layers" International Journal of Sensors (Basel), vol. 14, no. 1, 2014, pp. 795–833.

[4] M. Garcia, S. Sendra, M. Atenas, and J. Lloret, "Underwater Wireless Ad-hoc Networks: A Survey," International Journal of Mobile Ad hoc Networks: Current Status and Future Trends, 2011, pp. 379-411.

[5] K. Shazzad, T. Kemal, and A. Esam, "Multi-Hop-Enabled Energy-Efficient MAC Protocol for Underwater Acoustic Sensor Networks," International Journal of Sensor and Actuator Networks, vol. 4, no. 3, 2015, pp. 226-250.

[6] M. Ayaz, I. Baig, A. Abdullah, and I. Faye, "A Survey on Routing Techniques in Underwater Wireless Sensor Networks," Journal of Network and Computer Applications, vol. 3, no. 4, 2011, pp. 1908-1927.

[7] J. Lloret, M. Garcia, S. Sendra, and G. Lloret, "An underwater wireless group-based sensor network for marine fish farms sustainability monitoring," International Journal of Telecommunication Systems, 2014, pp. 1-18.

[8] M. Ayaz and A. Abdullah, "Underwater Wireless Sensor Networks: Routing Issues and Future Challenges," Proc. of the 7th International Conference on Advances in Mobile Computing and Multimedia. ACM, Malaysia, 2009, pp. 370-375.

[9] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges," Ad Hoc Networks (Elsevier), vol. 3, no. 3, Mar. 2005, pp. 257–279.

[10] D. Pompili, T. Melodia, and I. F. Akyildiz, "Distributed Routing Algorithms for Underwater Acoustic Sensor Networks," IEEE Trans. Wireless Communications, vol. 9, no. 9, Sept. 2010, pp. 2934-3944.

[11] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based Forwarding Protocol for Underwater Sensor Networks," International Conference on Networking (IFIP networking), 2006, pp. 1-20.

[12] N. Nicolaou, A. See, P. Xie, J.-H. Cui, and D. Maggiorini, "Improving the Robustness of Location-based Routing for Underwater Sensor Networks," Proc. Of the OCEANS'07, Europe, June 2007, pp. 1-6.

[13] A. Sharma and H. A. Gaafar, "A Survey on Routing Protocols for Underwater Sensor Networks," International Journal of Computer Science & Communication Networks,vol. 2, no. 1, 2012, pp. 74-82.

[14] P. Xie, Z. Zhou, Z. Peng, J.-H. Cui, and Z. Shi, "Void Avoidance in Three-dimentional Mobile Underwater Sensor Networks," Proc. of the 4th International Conference of Wireless Algorithms, System, and Applications (WASA 2009), USA, August 2009, pp. 305-314.

[15] W. Bo, L. Yong-mei, and J. Zhigang, "ES-VBF: An Energy Saving Routing Protocol," Proc. of the 2012 International Conference on Information Technology and Software Engineering, 2012, pp. 87-97.

[16] E. A. Carlson, P.P. Beaujean and E. An., "Location-aware Routing Protocol for Underwater Acoustic Networks," Proc. of the OCEANS, 2006, pp. 1-6.

[17] R. Thumpi, R.B. Manjula, and S.M. Sunilkumar, "A Survey on Routing Protocols for Underwater Acoustic Sensor Networks," International Journal of Recent Technology and Engineering (IJRTE), vol. 2, May 2013, pp. 170-175.

[18] NS-2: Network Simulator, http://www.isi.edu/nsnam/ns, last access: 30th August 2015.

[19] P. Xie, et al., "Aqua-sim: A Ns-2 based Simulator for Underwater Sensor Networks," Proc. of IEEE/MTS OCEANS, 2009, pp. 1-7.

[20] LinkQuest, http://www.link-quest.com, last access: 30th August 2015.

Analytic Method for Evaluation of the Weights of a Robust Large-Scale Multilayer Neural Network

Mikael Fridenfalk
Department of Game Design
Uppsala University
Visby, Sweden
mikael.fridenfalk@speldesign.uu.se

***Abstract*—The multilayer feedforward neural network is presently one of the most popular computational methods in computer science. However, the current method for the evaluation of its weights is performed by a relatively slow iterative method known as backpropagation. According to previous research on a large-scale neural network with many hidden nodes, attempts to use an analytic method for the evaluation of the weights by the linear least square method showed to accelerate the evaluation process significantly. Nevertheless, the evaluated network showed in preliminary tests to fail in robustness compared to well-trained networks by backpropagation, thus resembling overtrained networks. This paper presents the design and verification of a new method that solves the robustness issues for such a neural network, along with MATLAB code for the verification of key experiments.**

Keywords–analytic; big data; FNN; large-scale; least square method; multilayer; neural network; robust; sigmoid.

I. INTRODUCTION

As an extension of an earlier work presented in ADVCOMP 2014 [1], this paper reconfirms the initial inference by the presentation of a new layer of experiments. As a brief introduction, the artificial neural network constitutes one of the most useful and popular computational methods in computer science. The most well-known category is the multilayer Feedforward Neural Network, in this paper abbreviated as FNN, where the weights are estimated by an iterative training method called backpropagation [2], [3]. Although backpropagation is relatively fast for small networks, it is rather slow for large ones, given the computational power of modern computers [4], [5]. To accelerate the training speed of FNNs, many approaches have been suggested based on the least square method [6]. Although the presentation on the implementation, as well as of the data on the robustness of these methods may be improved, the application of the least square method seems to be a promising path to investigate [7], [8].

What we presume to be required for a new method to replace backpropagation in such networks, is not only that it is efficient, but also that it is superior compared to existing methods and is easy to understand and implement. Therefore, the goal of this work has been to investigate the possibility to find a *robust analytic solution* (i.e., with good generalization abilities compared with a well-trained network using backpropagation, but without any iterations involved), for the weights of an FNN, which is easily understood and may be implemented relatively effortlessly, using a mathematical application such as MATLAB [9].

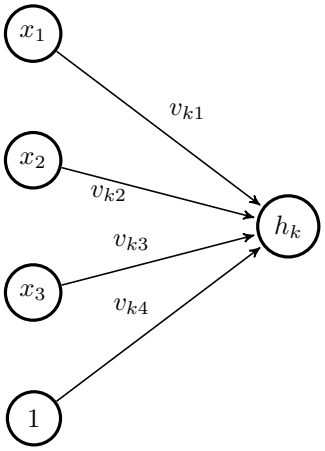


Figure 1. An example with three input nodes ($M = 3$), $h_k = S(\mathbf{v}_k\mathbf{u}) = S(v_{k1}x_1 + v_{k2}x_2 + v_{k3}x_3 + v_{k4})$, using a sigmoid activation function S.

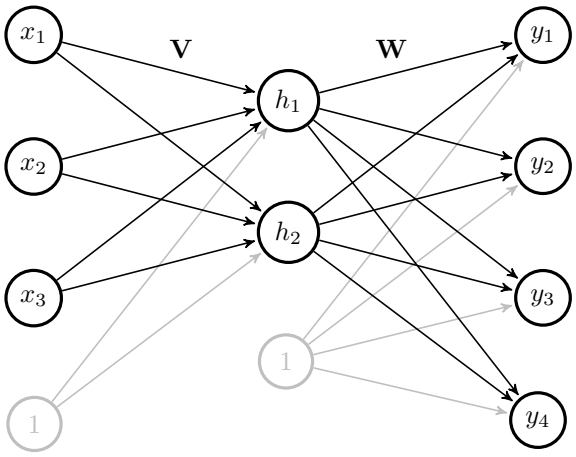


Figure 2. A vectorized model of a standard FNN with a single hidden layer, in this example with $M = 3$ input nodes, $H = 2$ hidden nodes, $K = 4$ output nodes and the weight matrices $\mathbf{V}$ and $\mathbf{W}$, using a sigmoid activation function for the output of each hidden node. In this model, the biases for the hidden layer and the output layer correspond to column $M + 1$ in $\mathbf{V}$ versus column $H + 1$ in $\mathbf{W}$.

II. OVERVIEW

As an overview of the main structure of this paper, in Section III, the history of artificial neural networks is briefly reiterated. In Sections IV-V a recap is made of the theory behind the fundamentals of the analytic method presented

in this paper. In Section VI, the derivation of an upgrade is reiterated for the improvement of the robustness of the proposed analytic method. In Section VII, the experimental setup is briefly described for the experiments presented in this paper, using a mathematical engine based on C++. In the results sections, Section VIII presents experiments for the evaluation of the original analytic solution we proposed in [1], in this paper labeled as the *initial experiments*. Section IX presents an evaluation of an upgrade of the original proposal, denoted as *diagonal reinforcement* [10], and labeled as the *primary experiments*. Section X presents a MATLAB-based version of the experiments presented in Figures 4-9, thereby additionally validating the experiments presented in Section IX, thus fulfilling the initial goal of this project, as formulated in [1], and the introduction section.

III. Background

The history of artificial neural networks is considered to have started in 1943 [11]. The 1950s and 1960s is often regarded as a golden age for neural networks, marked among other things by the development of the first successful neurocomputer, Mark I Perceptron [12]. However, towards the end of the 1960s, this age was turned into an ice age after the criticism of the perceptron model [13]. In the following decades the neural networks research slowly recovered by the introduction of multilayer networks and the introduction of backpropagation [12], [14]. Backpropagation is a slow iterative method for the evaluation of the weights of multilayer neural networks. During the last decades a large number of neural networks have been suggested, but presently the evaluation of the weights of the most well-known and widely used category, namely the FNN, is still based on backpropagation, which so far has made many mathematicians to avoid this field, since an analytic method for the evaluation of the weights of robust FNNs is considered today to be missing.

IV. Related Work

In a previous work [1], an analytic solution was proposed for the evaluation of the weights of a textbook FNN. This solution was found to be significantly much faster, and for $H = N - 1$ (where H denotes the number of hidden nodes and N, the number of training points), more accurate than solutions provided by backpropagation, but at the same time significantly less robust (e.g., more noise sensitive) compared to a well-trained network using backpropagation, why the analytic solution was in this context considered to lack robustness for direct use.

However, further experiments showed that even small measures, such as an increase in the input range of the network by the duplication of the training set, with the addition of perturbation, led to significant improvement of the robustness of the evaluated network. As a systematic attempt to address the issue of robustness, this paper presents the derivation, implementation and further verification of the new method proposed in [10], based on the expansion of the training set of an FNN, with addition of perturbation, but in practice without any impact on the execution speed of the original method introduced in [1].

V. An Analytic Solution

In [1], a textbook FNN is vectorized based on a sigmoid activation function $S(t) = 1/(1 + e^{-t})$. The weights $\mathbf{V}$ and $\mathbf{W}$ of such system (often denoted as W_{IH} versus W_{HO}), may be represented by Figures 1-2. In this representation, defined here as the normal form, the output of the network may be expressed as:

$$\mathbf{y} = \mathbf{W}\mathbf{h} = \mathbf{W}\begin{bmatrix} S(\mathbf{V}\mathbf{u}) \\ \hline 1 \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{1}$$

where $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_M]^T$ denotes the input signals, $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_K]^T$ the output signals, and S, an element-wise sigmoid function. In this paper, a winner-take-all classification model is used, where the final output of the network is the selection of the output node that has the highest value. Since the sigmoid function is constantly increasing and identical for each output node, it can be omitted from the output layer, as $\max(\mathbf{y})$ results in the same node selection as $\max(S(\mathbf{y}))$. Further on, presuming that the training set is highly fragmented (the input-output relations in the training sets were in our experiments established by a random number generator), the number of hidden nodes is in many experiments set to $H = N - 1$. Defining a batch of input signals, e.g., a training set, the input matrix $\mathbf{U}$ may be expressed as:

$$\mathbf{U} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{2}$$

where column vector i in $\mathbf{U}$, corresponds to training point i, column vector i in $\mathbf{Y}_0$ (target output value) and in $\mathbf{Y}$ (actual output value). Further, defining $\mathbf{H}$ of size $N \times N$, as the batch values for the hidden layer, given a training set of input and output values and $M^+ = M + 1$, the following relations hold:

$$\mathbf{U} = \begin{bmatrix} \mathbf{X} \\ \hline \mathbf{1}^T \end{bmatrix} : [M^+ \times N] \tag{3}$$

$$\mathbf{H} = \begin{bmatrix} S(\mathbf{V}\mathbf{U}) \\ \hline \mathbf{1}^T \end{bmatrix} : [N \times N] \tag{4}$$

$$\mathbf{Y} = \mathbf{W}\mathbf{H} : [K \times N] \tag{5}$$

To evaluate the weights of this network analytically, we need to evaluate the target values (points) of $\mathbf{H}_0$ for the hidden layer. In this context, the initial assumption is that any point is feasible, as long as it is unique for each training set. Therefore, in this model, $\mathbf{H}_0$ is composed of random numbers. Thus, the following evaluation scheme is suggested for the analytic solution of the weights of such network:

$$\mathbf{V}^T = (\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{H}_0^T : [M^+ \times H] \tag{6}$$

$$\mathbf{W}^T = (\mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}\mathbf{Y}_0^T : [N \times K] \tag{7}$$

where a linear least square solution is used for the evaluation of each network weight matrix. Such equation is nominally expressed as:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{8}$$

with the least square solution [6]:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{9}$$

Since the mathematical expressions for the analytic solution of the weights of a neural network may be difficult to

follow, an attempt has been made in Figure 3 to visualize the matrix operations involved. While a nonlinear activation function (such as the sigmoid function) is vital for the success of such network, the inclusion of a bias is not essential. It is for instance possible to omit the biases and to replace $\mathbf{H}_0$ with an identity matrix $\mathbf{I}$. Such a configuration would instead yield the following formula for the evaluation of $\mathbf{V}$ and $\mathbf{H}$ (where $\mathbf{UI}$ can further be simplified as $\mathbf{U}$):

$$\mathbf{V}^T = (\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{I} : [M^+ \times N] \tag{10}$$

$$\mathbf{H} = S(\mathbf{V}\mathbf{U}) : [H \times N] \tag{11}$$

VI. Diagonal Reinforcement

To recap the theory on diagonal reinforcement, as proposed in [10], we expand the input training set $\mathbf{U}$ in (2), by the addition of perturbation to the input signal, given the definition $\mathbf{\Theta} = \mathbf{U}\mathbf{U}^T$, with $\theta_{M^+M^+} = N$ in:

$$\mathbf{\Theta} = \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1M} & \theta_{1M^+} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2M} & \theta_{2M^+} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \theta_{M1} & \theta_{M2} & \dots & \theta_{MM} & \theta_{MM^+} \\ \theta_{M^+1} & \theta_{M^+2} & \dots & \theta_{M^+M} & N \end{bmatrix} \tag{12}$$

Further, an extended matrix $\tilde{\mathbf{U}}$ is introduced, where:

$$\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{\mathbf{U}}_1 & \tilde{\mathbf{U}}_2 & \dots & \tilde{\mathbf{U}}_N \end{bmatrix} \tag{13}$$

with:

$$\mathbf{U}_j = \begin{bmatrix} u_{1j}+\Delta & u_{1j}-\Delta & u_{1j} & u_{1j} & \cdots & u_{1j} & u_{1j} \\ u_{2j} & u_{2j} & u_{2j}+\Delta & u_{2j}-\Delta & \cdots & u_{2j} & u_{2j} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{Mj} & u_{Mj} & u_{Mj} & u_{Mj} & \cdots & u_{Mj}+\Delta & u_{Mj}-\Delta \\ 1 & 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \tag{14}$$

and where Δ is defined as the amplitude of the perturbation. Thus, for the right hand side of (8), $\tilde{\mathbf{\Theta}} = \tilde{\mathbf{U}}\mathbf{U}^T$, or more explicitly:

$$\tilde{\mathbf{\Theta}} = 2M \begin{bmatrix} d_1 & \theta_{12} & \dots & \theta_{1M} & \theta_{1M^+} \\ \theta_{21} & d_2 & \dots & \theta_{2M} & \theta_{2M^+} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \theta_{M1} & \theta_{M2} & \dots & d_M & \theta_{MM^+} \\ \theta_{M^+1} & \theta_{M^+2} & \dots & \theta_{M^+M} & N \end{bmatrix} \tag{15}$$

with $d_i = \theta_{ii} + \alpha$, where $\alpha = N\Delta^2/M$, or:

$$\tilde{\mathbf{\Theta}} = 2M\left[\mathbf{\Theta} + \mathrm{diag}(\alpha, \alpha, \dots, \alpha, 0)\right] \tag{16}$$

where $\mathrm{diag}(d_1, d_2, \dots, d_{M^+})$ denotes a diagonal matrix of size $M^+ \times M^+$ (where $M^+ = M+1$), with the diagonal elements $d_1, d_2, \dots, d_{M^+}$. Similarly, for the left hand side of (8), $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ are defined as:

$$\mathbf{H}_0^T = \mathbf{\Psi} = \begin{bmatrix} \psi_{11} & \psi_{12} & \dots & \psi_{1H} \\ \psi_{21} & \psi_{22} & \dots & \psi_{2H} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{N1} & \psi_{N2} & \dots & \psi_{NH} \end{bmatrix} \tag{17}$$

$$\mathbf{\Lambda} = \mathbf{U}\mathbf{H}_0^T = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1H} \\ \lambda_{21} & \lambda_{22} & \dots & \lambda_{2H} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{M^+1} & \lambda_{M^+2} & \dots & \lambda_{M^+H} \end{bmatrix} \tag{18}$$

and thereby, $\mathbf{\Psi}$ and $\mathbf{\Psi}_j$ as:

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{\Psi}_1 \\ \mathbf{\Psi}_2 \\ \vdots \\ \mathbf{\Psi}_N \end{bmatrix} \tag{19}$$

$$\mathbf{\Psi}_j = \begin{bmatrix} \psi_{j1} & \psi_{j2} & \dots & \psi_{jH} \\ \psi_{j1} & \psi_{j2} & \dots & \psi_{jH} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{j1} & \psi_{j2} & \dots & \psi_{jH} \end{bmatrix} \tag{20}$$

with $\tilde{\mathbf{\Lambda}} = \tilde{\mathbf{U}}\mathbf{\Psi}$:

$$\tilde{\mathbf{\Lambda}} = 2M\mathbf{\Lambda} \tag{21}$$

This transforms (8) into:

$$\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{X} = \tilde{\mathbf{U}}\mathbf{\Psi} \tag{22}$$

or:

$$\tilde{\mathbf{\Theta}}\mathbf{X} = \tilde{\mathbf{\Lambda}} \tag{23}$$

Thus:

$$2M\left[\mathbf{\Theta} + \mathrm{diag}(\alpha, \alpha, \dots, \alpha, 0)\right]\mathbf{X} = 2M\mathbf{\Lambda} \tag{24}$$

Given the matrix equation:

$$\mathbf{A}\mathbf{X} = \mathbf{B} \tag{25}$$

since, given a scalar $c \in \mathbb{R}$:

$$c \cdot (\mathbf{A}\mathbf{X}) = (c \cdot \mathbf{A})\mathbf{X} = c \cdot \mathbf{B} \tag{26}$$

thereby:

$$\left[\mathbf{\Theta} + \mathrm{diag}(\alpha, \alpha, \dots, \alpha, 0)\right]\mathbf{X} = \mathbf{\Lambda} \tag{27}$$

This yields thus, the final expression:

$$\left[\mathbf{U}\mathbf{U}^T + \mathrm{diag}(\alpha, \alpha, \dots, \alpha, 0)\right]\mathbf{X} = \mathbf{U}\mathbf{H}_0^T \tag{28}$$

Hence, the expansion of $\mathbf{U}$ into a perturbation matrix $\tilde{\mathbf{U}}$ of size $M^+ \times 2MN$, and similarly of $\mathbf{H}_0^T$ into a matrix $\mathbf{\Psi}$ of size $2MN \times H$, is according to (28) equivalent to the reinforcement of the diagonal elements of the square matrix $\mathbf{\Theta} = \mathbf{U}\mathbf{U}^T$, by the addition of a scalar $\alpha = N\Delta^2/M$ to each diagonal element, except for the last one, which as a consequence of the use of bias in the network, is left intact.

VII. Experimental Setup

The experiments presented in Sections VIII-IX, are based on a minimal mathematical engine that was developed in C++, with the capability to solve $\mathbf{X}$ in a linear matrix equation system of the form $\mathbf{A}\mathbf{X} = \mathbf{B}$, where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{X}$ denote matrices of appropriate sizes, since it is computationally more efficient to solve a linear equation system directly, than by matrix inversion. In this system, the column vectors of $\mathbf{X}$ are evaluated using a single Gauss-Jordan elimination cycle [6], where each column vector $\mathbf{x}_i$ in $\mathbf{X}$ corresponds to the column vector $\mathbf{b}_i$ in $\mathbf{B}$. Backpropagation was in these experiments, for high execution speed (and a fair comparison with the new methods), also implemented in C++, using the code

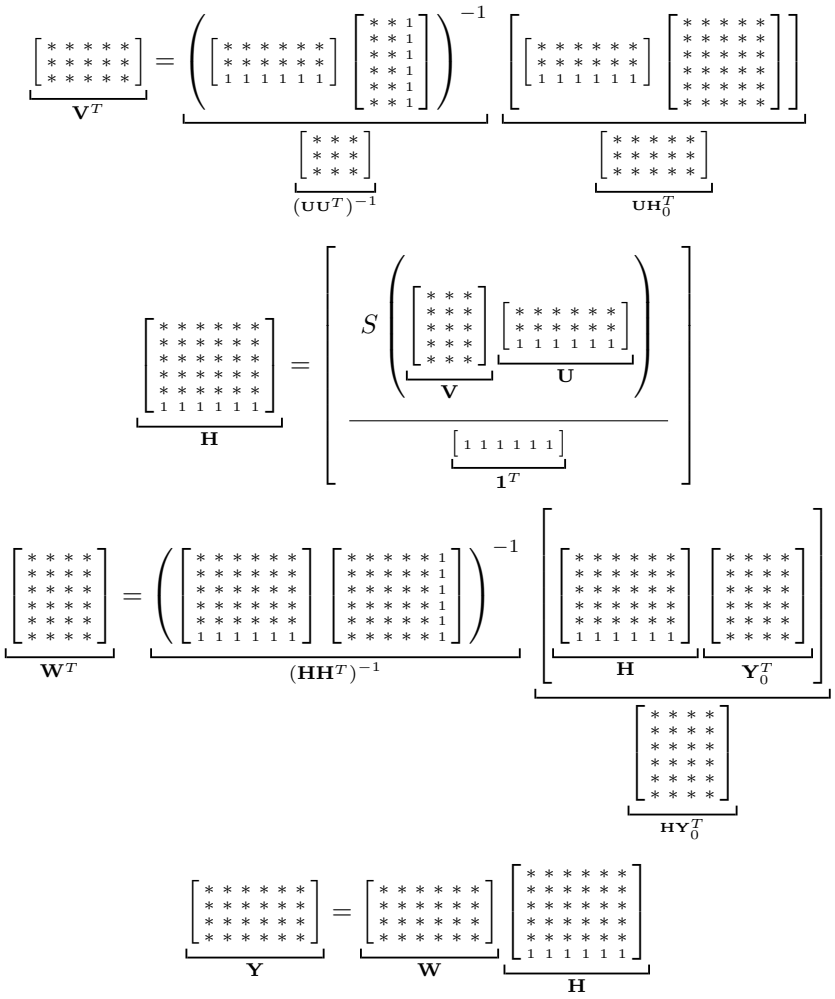


Figure 3. A visual representation of the evaluation of weights $\mathbf{V}$ and $\mathbf{W}$ by the analytic method presented in the initial work [1], and the actual output $\mathbf{Y}$, in this example as a function of six training points, $N = 6$, the training input and output sets $\mathbf{U}$ (for a simplified notation, whenever the relation $\mathbf{U} = \mathbf{U}_0$ is implicit) and $\mathbf{Y}_0$, with two inputs, $M = 2$, four outputs, $K = 4$, and five hidden nodes, $H = N - 1 = 5$. In this figure, an asterisk denotes a floating-point number. To facilitate bias values, certain matrix elements are set to one.

presented in [15] as a reference. To measure the efficiency of the new method in [1], compared with the standard method (backpropagation), the standard textbook definition for the mean-squared error was used:

$$\epsilon = \frac{1}{2N} \sum_{i}^{K} \sum_{j}^{N} (a_{ij} - y_{ij})^2 \tag{29}$$

where a_{ij} denotes an element in $\mathbf{Y}_0$ (target value), and y_{ij} the corresponding element in $\mathbf{Y}$ (actual value). Although the new method, as in [1], does not intrinsically benefit from such definition (since there is no need here for the differentiation of the mean-squared error, which however, is essential for backpropagation), to simplify comparison in Tables I-V, the same definition of the mean-squared error was also applied to the new method.

VIII. Initial Experiments

The experimental results presented in this section, as shown in Tables I-V, are based on ten individual experiments for each parameter setting using different random seeds, where $\bar{t}$ denotes average execution time, using a single CPU core on a modern laptop computer (the same computer was used for all experiments presented in this paper), $\bar{\epsilon}$, the average value of the mean-squared errors, and $\tilde{\epsilon}$, the median value. The success rate, $\bar{s}$, is similarly based on an average value. Since the variation of the results is large between the experiments, the average values are in general larger than the median values. If the number of experiments per parameter setting is increased, the average value tends to increase as well.

On a note of preliminary experiments with respect to robustness, regarding the generalization abilities of the network, the original method (i.e., without the application of diagonal reinforcement) showed to steeply lose accuracy with the addition of noise to the input values, compared with a network trained by backpropagation. This shows that although the results seem to be in order according to Tables I-V, the original method lacks robustness for direct use. However, further experiments showed that even small measures, such as an increase in the input range of the network by the duplication

Table I. Initial experiments (Tables I-V) – Backpropagation with $H = N - 1$ and average success rate $\bar{s}$.

M	N	H	K	Iterations	$\bar{t}$	$\bar{\epsilon}$	$\tilde{\epsilon}$	$\bar{s}$ (%)
5	20	19	5	10^4	46.6 ms	0.0671	0.0620	93.0
5	20	19	5	10^6	4.39 s	0.0175	$4.64 \cdot 10^{-5}$	96.5
10	50	49	10	10^4	182 ms	0.116	0.114	83.2
10	50	49	10	10^6	18.1 s	0.0680	0.0650	86.4
20	50	49	20	10^4	333 ms	0.0394	0.0392	94.6
20	50	49	20	10^6	33.3 s	0.0170	0.0200	96.6
40	100	99	40	10^4	1.27 s	0.0671	0.0670	88.9
40	100	99	40	10^6	127 s	0.0180	0.0200	96.4

Table II. New method with $H = N - 1$.

M	N	H	K	$\bar{t}$	$\bar{\epsilon}$	$\tilde{\epsilon}$	$\bar{s}$ (%)
5	20	19	5	332 μs	$3.34 \cdot 10^{-8}$	$2.00 \cdot 10^{-13}$	100.0
10	50	49	10	3.74 ms	$6.99 \cdot 10^{-9}$	$2.26 \cdot 10^{-12}$	100.0
20	50	49	20	4.79 ms	$2.68 \cdot 10^{-13}$	$5.93 \cdot 10^{-16}$	100.0
40	100	99	40	36.7 ms	$3.93 \cdot 10^{-12}$	$2.33 \cdot 10^{-13}$	100.0

Table III. Backpropagation with $H < N - 1$ and 10^4 iterations.

M	N	H	K	$\bar{t}$	$\bar{\epsilon}$	$\tilde{\epsilon}$	$\bar{s}$ (%)
5	20	5	5	14.7 ms	0.130	0.125	86.5
10	50	10	10	43.1 ms	0.227	0.237	71.6
20	50	20	20	144 ms	0.0624	0.0599	94.2
40	100	40	40	531 ms	0.115	0.115	84.8

Table IV. New method with $H < N - 1$.

M	N	H	K	$\bar{t}$	$\bar{\epsilon}$	$\tilde{\epsilon}$	$\bar{s}$ (%)
5	20	5	5	59 μs	0.281	0.289	58.5
10	50	10	10	370 μs	0.350	0.350	50.0
20	50	20	20	1.30 ms	0.279	0.277	91.8
40	100	40	40	9.24 ms	0.290	0.290	98.5

Table V. Selective use of bias for the new method, with $M = 40$, $N = 100$, $H = 99$, and $K = 40$.

H_b	Y_b	$\bar{t}$	$\bar{\epsilon}$	$\tilde{\epsilon}$	$\bar{s}$ (%)
No	No	35.7 ms	0.00514	0.00513	100.0
No	Yes	36.4 ms	$1.35 \cdot 10^{-12}$	$1.91 \cdot 10^{-14}$	100.0
Yes	No	36.2 ms	0.00544	0.00534	100.0

Table VI. Average execution time, $\bar{t}_{\text{bp}}$ (backprop.), $\bar{t}_{\text{new}}$ (initial), $\bar{t}_{\text{new+}}$ (primary), $\bar{t}_{\text{new*}}$ (verification).

Figure	M	N	H	K	$\bar{t}_{\text{bp}}$	$\bar{t}_{\text{new}}$	$\bar{t}_{\text{new+}}$	$\bar{t}_{\text{new*}}$
4	10	25	24	10	92.5 ms	688 μs	668 μs	326 μs
5	20	50	49	20	332 ms	4.84 ms	4.86 ms	559 μs
6	40	100	99	40	1.27 s	37.0 ms	37.1 ms	1.47 ms
7	10	25	10	10	43.3 ms	212 μs	202 μs	175 μs
8	20	50	20	20	144 ms	1.33 ms	1.31 ms	306 μs
9	40	100	40	40	535 ms	9.35 ms	9.38 ms	631 μs

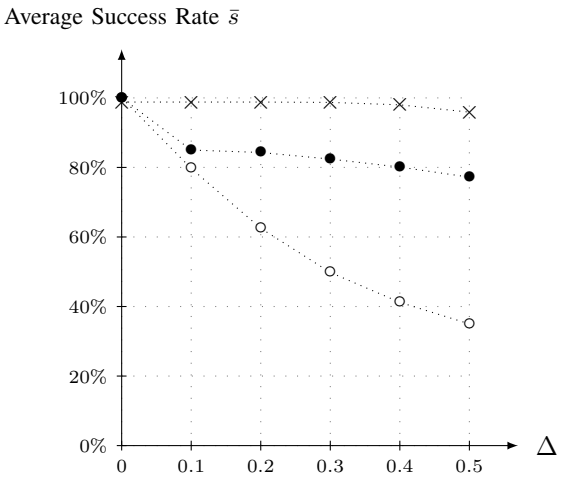


Figure 4. Backpropagation (×), initial analytic method (○), versus new method using diagonal reinforcement (●), with $M = 10$ (input nodes), $N = 25$ (training points), $H = 24$ (hidden nodes), and $K = 10$ (output nodes).

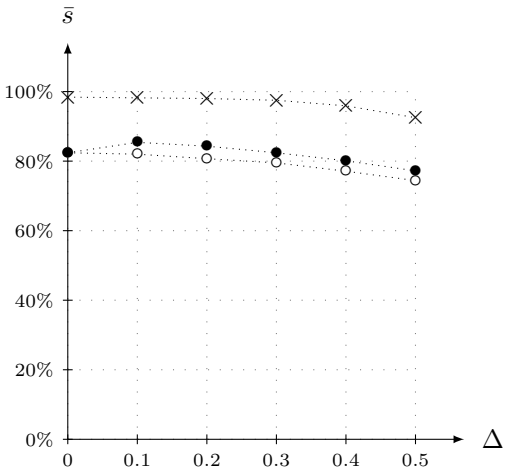


Figure 7. $M = 10$, $N = 25$, $H = 10$, and $K = 10$.

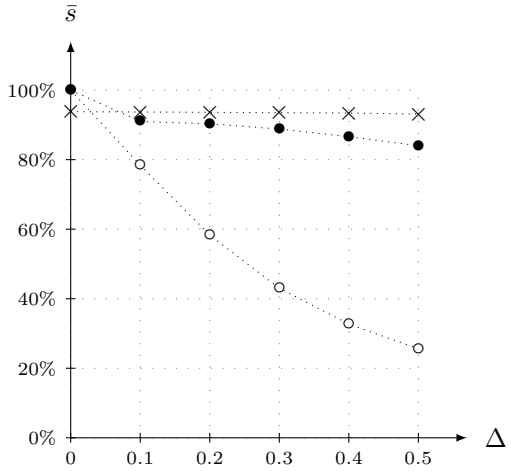


Figure 5. $M = 20$, $N = 50$, $H = 49$, and $K = 20$.

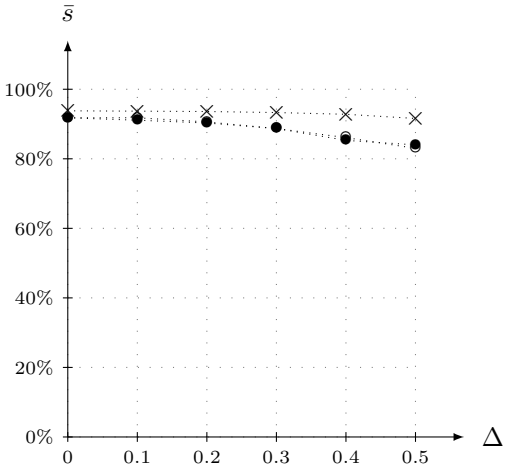


Figure 8. $M = 20$, $N = 50$, $H = 20$, and $K = 20$.

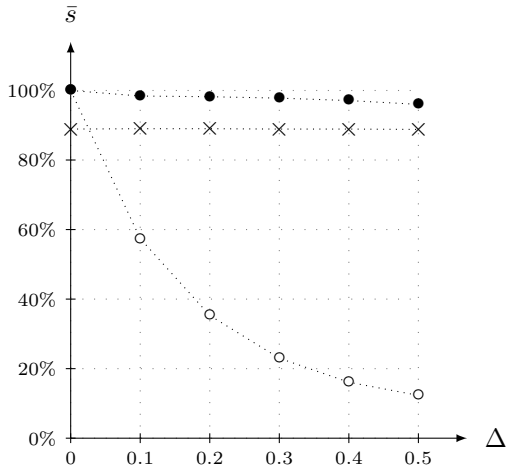


Figure 6. $M = 40$, $N = 100$, $H = 99$, and $K = 40$.

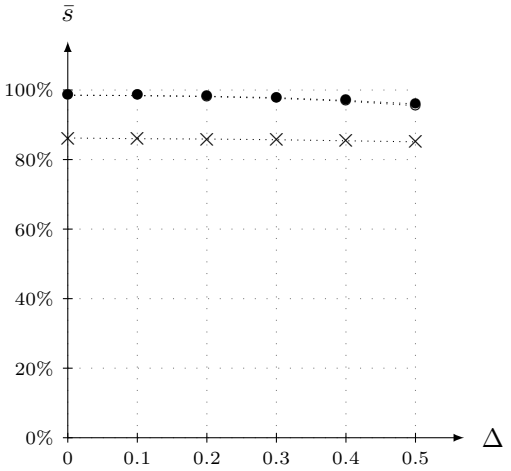


Figure 9. $M = 40$, $N = 100$, $H = 40$, and $K = 40$.

of the training set with the addition of perturbation and a more conscious design of $\mathbf{H}_0$, by for instance the clustering of the random values as a function of the output values, or for layers with few hidden nodes, the binary encoding [16] of $\mathbf{H}_0$, led to significant improvements of the robustness of the new method. Therefore, this was an indication at an early stage of the development of the method presented in this paper, that these robustness issues could be solved, and in this context, without any significant impact to the computational speed of the initial version of the new method (i.e., without diagonal reinforcement).

IX. Primary Experiments

The experimental results presented in this section, as shown in Figures 4-9, are in similarity with previous section based on a mathematical engine developed in C++, here in addition examining the effects of diagonal reinforcement, as described in Section VI, measuring average success rate, and Table VI, measuring execution speed. Each experiment is based on ten individual experiments (with different random seeds), using a single CPU core. In Table VI, $\bar{t}_{\text{bp}}$ denotes the execution time for backpropagation based on 10000 iterations, which applies to all backpropagation experiments presented in this paper. Similarly, $\bar{t}_{\text{new}}$ denotes the execution time for the original analytic method in [1], and $\bar{t}_{\text{new+}}$, the execution time for the new method, using diagonal reinforcement.

In all the experiments presented in this paper, the input values to the FNN is based on the integers $\{0, 1, 2\}$, and the output values of a binary number, $\{0, 1\}$. To avoid inconsistencies (or repetition) in any training set, no identical input vectors are permitted. For the addition of noise, a random value (with uniform distribution) in the range of $\pm\Delta$, was added to each input value. However, although according to our derivation of (28), $\alpha = N\Delta^2/M$, α had in practice to be retuned to $10^5 \cdot N\Delta^2$ for good results in the experiments in Figures 4-6 ($H = N - 1$), and to $10^4 \cdot N\Delta^2$ in Figures 7-9 (few hidden nodes).

X. Matlab Verification Experiments

This paper presents a verification of previously presented results in [10], but here based on MATLAB. As initially inferred in [1], the goal in this work is to develop a method that is efficient, superior compared to existing methods, and in addition easy to understand and implement using a mathematical application such as MATLAB.

Thus, to verify the results presented in Figures 4-9, which were based on our own computational engine developed in C++, we hereby present a MATLAB version of the same solution, as presented in Figure 10 (and verified by the auxiliary MATLAB code in Figures 11-13), corresponding to the same experiments as in Figures 4-9.

The MATLAB code in Figures 10-13, produces numerical results that coincide relatively accurately (considering the use of random numbers) with previous evaluations using the engine based on C++. The execution speed proved to be faster, as shown in Table VI, compared to the C++ engine. According to this table, while for smaller matrix sizes, the execution speed is only marginally faster for MATLAB (defined by execution time, $\bar{t}_{\text{new*}}$), in comparison with the C++ engine ($\bar{t}_{\text{new+}}$), however, for larger matrices, MATLAB is significantly faster. One reason is that while MATLAB is multithreaded, the C++ engine uses only a single thread. In addition, while the core computational engine of MATLAB is expected to be implemented by assembly code, we used regular C++ code, prioritizing code readability. MATLAB is in addition expected to use smart tricks to accelerate matrix operations, including using CPU cores very efficiently.

However, as a note, although MATLAB in average showed to be faster than our C++ engine, the application proved to be initially slower at a startup phase, running the MATLAB master script in Figure 13. This script is thus, for a more representative time measurement, recommended to be executed twice.

Since the MATLAB code presented in this paper constitutes the actual results of this work, we have below included a brief explanation of this code. To facilitate direct copy and paste of this code from a PDF version of this document into the suggested filenames (e.g., EvalVW.m for the code in Figure 10), the code have been placed in a single column environment and without line numbers. In addition, on two instances, a comma is placed after an end-statement, since copy and paste may place two end-statements after one another, which presently generates an error in MATLAB. The commas are not visible in the code (due to white font color), but appear after a paste operation into an m-file.

A. EvalVW.m

The function EvalVW constitutes the core results of this paper. The MATLAB function `rand(h,n)` creates a matrix of size $H \times N$, consisting of random numbers within the interval of 0 to 1. Regarding the evaluation of $\mathbf{Q}$, note that only the first M diagonal elements are affected by the application of diagonal reinforcement. In the evaluation of $\mathbf{W}$, since $\mathbf{H}\mathbf{H}^T$ showed in our experiments to often be a near singular matrix, MATLAB will by default issue a warning statement at execution, unless the warning messages are temporarily turned off and on again by the commands `warning('off','all')` and `warning('on','all')`.

B. GenTrainingSet.m

This function calculates a training set, i.e., a complete set of input and output signals for the training of an FNN. The generation of the input matrix $\mathbf{U}_0$ is straightforward and concise, yielding unique columns of values in $\{0, 1, 2\}$, based on random numbers. As a note on the generation of $\mathbf{Y}_0$, which is also based on random numbers, this loop-based solution showed to be both straightforward and fast.

C. Exp.m

This inner loop experimental function is used for the complete evaluation of a single point in Figures 4-9. As shown here, N1 denotes the number of FNNs that are generated and tested, and N2, the number of input/output tests performed on each evaluated FNN. On time measurements, since we are only interested in the time it takes to evaluate the weights $\mathbf{V}$ and $\mathbf{W}$, the `tic` (reset and start) and `toc` (stop) MATLAB timer commands, are only placed around the function EvalVW.

To obtain the exact same results each time the function Exp is called, using the same input parameters, the function `rng` is used to set the seed of the random number generator in MATLAB. This is actually not necessary at this stage of the

code development, but is often useful at a development stage of a system based on random numbers.

As a potential pitfall, note that the disturbance added to $\mathbf{U}$, is only allowed to affect the first M rows of this matrix (i.e., the input signals), and thus not the bias row, which by definition is always set to a column vector of ones.

The logical expression I0 == I, yields finally a value of one, for each element of I0 that is equal to the same element in I, else zero. This expression is used for the evaluation of the success rate of the network, by comparing the expected output $\mathbf{Y}_0$, with the actual output $\mathbf{Y}$.

D. NNX.m

The master script NNX.m (corresponding to the verification experiments presented in this paper) generates results that correspond to Figures 4-9, except for backpropagation, which was not implemented in MATLAB, but only in C++, since while matrix operations are efficient in MATLAB, the handling of iterations and scalars are, as a general rule, slower than C++.

XI. Discussion

The strength of an analytic method is clear in the sense that it could potentially expand the use of FNNs to a wider range of applications. However, at this stage, there are also some question marks that could require further study.

The first of these is the question of the use of diagonal reinforcement, compared with the initial method in [1], since as shown in Figures 4-9, almost the same effect can be produced by the reduction of the number of hidden nodes. From this perspective, diagonal reinforcement serves mainly to generalize the initial method as proposed in [1].

In addition, two observations that caught our attention during the experiments were that the matrix $\mathbf{H}\mathbf{H}^T$ in many cases was nearly singular in the evaluation of $\mathbf{W}$, and that α had to be set to a very high value in comparison to $N\Delta^2/M$, to have the intended effect. Thus, although the analytic method presented in this paper seems to operate correctly for large-scale FNNs, further analysis is recommended to shed light on its inner workings.

XII. Conclusion

The method presented in this paper provides for a robust analytic solution of the weights of a large-scale FNN (i.e., as previously defined, with good generalization abilities compared with a well-trained network using backpropagation, but without any iterations involved), which is significantly faster compared with backpropagation, yet straightforward to implement, using a mathematical application such as MATLAB.

Examples of a few application areas that could benefit from this method are artificial intelligence, economics, control theory, and in general any field dealing with big data for decision making, such as cancer treatment and research.

References

[1] M. Fridenfalk, "The Development and Analysis of Analytic Method as Alternative for Backpropagation in Large-Scale Multilayer Neural Networks," in ADVCOMP 2014: The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences, Rome, Italy, August 2014, pp. 46–49.

[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. 3rd ed., Prentice Hall, 2009, pp. 727–736.

[3] B. J. Wythoff, "Backpropagation Neural Networks: A Tutorial," in Chemometrics and Intelligent Laboratory Systems, vol. 18, no. 2, 1993, pp. 115–155.

[4] P. D. Wilde, Neural Networks Models: An Analysis. Springer, 1996, pp. 35–51.

[5] R. P. W. Duin, "Learned from Neural Networks," in ASCI 2000, Lommel, Belgium, 2000, pp. 9–13.

[6] C. H. Edwards and D. E. Penney, Elementary Linear Algebra. Prentice Hall, 1988, pp. 220–227.

[7] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," in Proceedings of the IEEE, vol. 78, no. 9, 1990, pp. 1415–1442.

[8] Y. Yam, "Accelerated Training Algorithm for Feedforward Neural Networks Based on Least Squares Method," in Neural Processing Letters, vol. 2, no. 4, 1995, pp. 20–25.

[9] "MATLAB, The MathWorks, Inc." 2015, URL: http://www.mathworks.com/ [accessed: 2015-11-24].

[10] M. Fridenfalk, "Method for Analytic Evaluation of the Weights of a Robust Large-Scale Multilayer Neural Network with Many Hidden Nodes," in ICSEA 2014: The Proceedings of the Ninth International Conference on Software Engineering Advances, Nice, France, October 2014, pp. 374–378.

[11] W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," in Bulletin of Mathematical Biophysics, vol. 5, 1943, pp. 115–133.

[12] N. Yadav, A. Yadav, and M. Kumar, "History of Neural Networks," in An Introduction to Neural Network Methods for Differential Equations. Springer, 2015, pp. 13–15.

[13] M. Minsky and S. Papert, Perceptrons. MIT Press, Cambridge, 1969.

[14] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation, Harvard University, 1974.

[15] M. T. Jones, AI Application Programming. 2nd ed., Charles River, 2005, pp. 165–204.

[16] F. Gray, "Pulse Code Communication," U.S. Patent no. 2 632 058, 1947.

```
function [V,W] = EvalVW(U0,Y0,h,d)
[mp,n] = size(U0);
H0 = rand(h,n);
Q = U0 * U0' + diag([d * ones(1,mp-1) 0]);
V = (Q\(U0 * H0'))';
H = [1./(1 + exp(-V*U0)); ones(1,n)];
warning('off','all');
W = ((H * H')\(H * Y0'))';
warning('on','all');
```

Figure 10. The results of this paper, condensed as the MATLAB function EvalVW.m.

```
function [U0,Y0] = GenTrainingSet(m,n,k)
ok = 0;
for i = 1:1000,
    UX = randi([0 2],floor(1.5*n),m);
    U0 = unique(UX,'rows','stable')';
    if size(U0,2) >= n, ok = 1; break; end
end
if ok, U0 = U0(:,1:n); U0 = [U0; ones(1,n)];
else U0 = -1; end
Y0 = zeros(k,n);
v = randi([1 k],1,n);
for i = 1:n, Y0(v(i),i) = 1; end
```

Figure 11. Generation of a training set for testing, GenTrainingSet.m.

```
function [sr,time] = Exp(m,n,h,k,d,delta)
N1 = 10; N2 = 100; srsum = 0; t = 0;
for j = 1:N1
    rng(1000*j);
    [U0,Y0] = GenTrainingSet(m,n,k);

    tic; [V,W] = EvalVW(U0,Y0,h,d); t = t + toc;

    [~,I0] = max(Y0);
    for i = 1:N2
        rng(1000*j+i);
        U = U0;
        U = U + [2 * delta * (rand(m,n) - .5); zeros(1,n)];
        H = [1./(1 + exp(-V*U)); ones(1,n)];
        Y = W * H; [~,I] = max(Y);
        srsum = srsum + 100 * sum(I == I0)/n;
    end
end
time = t/N1;
sr = srsum/(N1*N2);
```

Figure 12. Inner loop experiments Exp.m, called by NNX.m, based on 10 training sets, each performing 100 input/output tests.

```
clear, clc
m = [10 20 40]';
p = [m 2.5*m 2.5*m-1 m; m 2.5*m m m]
diagReinfRel = 10^4*[10 10 10 1 1 1];
for r = 1:6
    time = 0;
    m = p(r,1); n = p(r,2); h = p(r,3); k = p(r,4);
    dr = diagReinfRel(r) * n;
    for j = 1:2
        for i = 1:6
            delta = .1*(i-1);
            if j == 1, d = 0; else d = dr * delta * delta; end
            [sr,t] = Exp(m,n,h,k,d,delta);
            successRate(i,j) = sr;
            time = time + t;
        end
    end
    successRate
    aveTime(r) = time/12;
end
1000 * aveTime %milliseconds
```

Figure 13. The MATLAB master script NNX.m, the outer loop for the validation of the function EvalVW($\mathbf{U}_0, \mathbf{Y}_0, h, d$) in Figure 10, and the experimental results in Figures 4-9 (backpropagation excluded).

```
p =

    10    25    24    10
    20    50    49    20
    40   100    99    40
    10    25    10    10
    20    50    20    20
    40   100    40    40

ans =

  100.0000   78.7560   58.7600   46.2760   38.2480   32.4600
  100.0000   83.9840   82.4880   79.7600   76.4720   72.9680

ans =

  100.0000   68.1140   43.2180   30.1940   22.9300   18.5740
  100.0000   90.1140   88.9560   87.2680   85.0620   82.2980

ans =

  100.0000   75.1150   51.6290   35.2760   24.9340   18.6990
  100.0000   98.2980   97.9940   97.4470   96.5440   95.1400

ans =

   83.2000   82.2400   80.3280   77.6960   74.3560   70.7360
   83.2000   84.4800   82.6160   79.7920   76.4280   72.5720

ans =

   91.4000   90.4880   89.0460   86.8360   83.8920   80.4020
   91.4000   90.0060   88.9680   87.2620   85.0540   81.0420

ans =

   98.5000   98.2320   97.8330   97.1770   96.0400   94.3460
   98.5000   98.3130   98.0030   97.4440   96.5250   95.0490

ans =

    0.3257    0.5587    1.4664    0.1749    0.3056    0.6307
```

Figure 14. Results from the execution of NNX.m by MATLAB (with some line breaks removed), corresponding to the code in Figure 13, Figures 4-9 (C++), and the verification column in Table VI.

A Ludic/Narrative Interpretation of the Willingness to Repeatedly Play Mobile Serious Games

Alfredo Imbellone, Brunella Botte, Giada Marinensi, Carlo Maria Medaglia
Digital Administration and Social Innovation Center – DASIC, Link Campus University
Rome, Italy
a.imbellone@unilink.it, b.botte@unilink.it, g.marinensi@unilink.it, c.medaglia@unilink.it

***Abstract* — Mobile game-based learning is a very promising sector for corporate training, but it still lacks of a robust scientific research investigating the underlying causal models to explain its working principles. The paper presents the results of an empirical study conducted on two different kits of mobile serious games, composed respectively by 30 and 20 games, both developed in the framework of an European project on mobile game-based learning for corporate training, titled "InTouch". The study analyzes the causal relationships that influence the players' willingness to play again the serious games and interprets those findings according to the distinction between ludic and narrative components of the games. The results emerging from the testing of the games on two separate samples of 54 and 118 people can be interpreted as an empirical evidence of the simultaneous, and yet independent, significant role of both the ludic and narrative component of a serious game in determining the willingness to play again. On the whole, for the considered games the ludic factor showed to have a stronger influence than the narrative one. This different weight of ludic and narrative components, however, can be interpreted as a consequence of the specific mobile serious games that were analyzed, and cannot be generalized to other game-based solutions. Furthermore, when analyzing different groups within the considered sample, males and younger people showed to be more influenced by the ludic component, while the narrative component resulted to be stronger than the ludic one for females and older people. The conclusion of the study suggests an implementation of the proposed research concept with other serious games to find if the ludic/narrative interpretative key can improve understanding the structure of the games.**

Keywords - Mobile Game-Based Learning; Corporate Training; Serious Games; Ludology; Narratology

I. INTRODUCTION

In the frame of two editions of an European project about mobile Game-Based Learning (mGBL) in the period 2010-2015 two different kits of serious games were developed and tested for a total of 50 games [1].

The analysis of the kit of serious games for mobile devices is here referred to a subject of debate about games, concerning the relationship between narrative and game design, namely between ludology and narratology [2].

Mobile game-based learning is an educational trend that is gaining more and more in popularity. Its main advantages are considered mobility and portability, flexibility, accessibility, and informality [3].

Thanks to mGBL, didactic contents are made available anytime and anywhere, and learning is linked to activities in the outside world environment [4][5]. Serious games for mobile devices can teach soft skills that support self-efficacy, self-directed learning, and reflection upon performance [6][7].

Research on game-base learning and serious games is rapidly growing in the last years [8], but there are still few specific validated instruments of analysis [9][10][11] and there is a scarcity of studies strongly based on causal relationships [12][13].

The debate between narratology and ludology can be dated approximately around the second half of the 1990s, reaching its climax in the 2000s, to definitively find a reconciliation in these very last years.

The narratological position considers games as novel forms of narrative that must therefore be studied using theories of narrative.

Ludologists, on the other hand, state that games are essentially formal, contrary to narratives that are basically interpretative [14].

Games according to narratologists are closely related to narrative and stories: even thought basically made of rules, they mainly tell stories, contain narrative elements, and show narrative structural sequences [15].

Ludologists think that the study of games should concern the analysis of the abstract and formal systems they describe, that is game structure, rules, interactivity and gameplay. These are the elements that give immersion and the feel of real experience of a game and are more important than optional narrative elements [16].

Other hybrid approaches emerged trying to conciliate and comprehend both points of view.

Ryan proposed to incorporate narratology inside ludology, since it deals with the construction of stories that is similar to the game mechanics [17].

Aarseth, although considered a radical ludologist, stated that games and narrative significantly overlap [18].

Lindley unified in a heuristic triangular space ludology, narratology, and simulation, describing the relationships between gameplay and narrative as a competition determining ludic interaction on one side, and narrative patterns perception on the other side [19].

Jenkins proposed a middle-ground position, talking about games as "spaces" with narrative possibility enriching gameplay [20].

The present study aims at giving an empirical contribution to the debate among ludologists and narratologists, referring to it as an interpreting key for the results obtained from the study of some causal models derived from the analysis of two sets of serious games.

Points of interest of the present study can be considered: (1) the fact that it adds empirical data and analysis to a field that has been mainly developed on theoretical basis, (2) the study of causal relationships in the field of serious games with the use of Structural Equation Models (SEM), (3) the focus on serious games for mobile devices that represent an expanding sector [21].

In particular, the considered mobile serious games are very short in duration (few minutes to complete each game) and are playable through a touch-screen interface using only one finger. That is to say, they are games the users can play in short casual bursts of time, anywhere and at any time, at work or at home, or even on the way to/from work/home [22]. It has been considered relevant to transfer the ludology/narratology debate, usually referred to more structured games, to this kind of games.

This paper will give a description of the project, whose main objectives were the development and testing of the serious games kits (Section II). Scope and hypothesis of the present study will then be illustrated (Section III). Methods and results of the empirical analysis will be reported, illustrating the statistical work that has been done and what it produced (Sections IV and V). The paper will end with a conclusion and future work section (VI), explaining how the results of the present study can be interpreted in the light of the ludology/narratology debate, the limits of the present study, and how a further deepening of these issues can be addressed.

II. THE INTOUCH PROJECT

In November 2010 a consortium of European partners from eight countries (Sweden, Lithuania, United Kingdom, Italy, France, Austria, Bulgaria, and Switzerland) started working on "Labour Market InTouch: new non-routine skills via mobile game-based learning" project, funded by Leonardo da Vinci Multilateral Projects for Development of Innovation Program.

At the beginning of the project a field research was conducted in order to define the top 10 crucial transversal competences for non-routine tasks. 62 managers and employers of business service SMEs were interviewed in order to detect which were the most requested non-routine skills in labour market, and, for each skill, a list of associated situational cases.

A structured questionnaire formed by 51 items describing tasks/behaviors related to skill management was used to identify the most relevant non routine skills. The factorial analysis of gathered data highlighted ten factors, interpreted as the crucial non-routine skills: Communication, Planning, Conflict Management, Openness to change, Decision Making, Teamwork, Flexibility, Strategic thinking, Initiative, Learning and improvement. Thus, the InTouch project designed and developed 30 games for mobile devices (3 for each non routine skill) to enable adult workers to improve their ability to deal with non routine situations at work.

In the light of the good results achieved with this first kit of 30 serious games, in November 2013 the InTouch project was refunded by Leonardo da Vinci Transfer of Innovation Program, with the aim of addressing the specific learning needs of ICT SMEs employees. The second edition of the InTouch project, named InTouch-ICT for its focus on ICT SMEs, conducted a new field research through surveying 238 respondents in four countries: Turkey, Hungary, Italy, and Sweden. In this case, people were directly faced with a list of 15 skills, asking to express the degree of importance for each one of them. According with the results of the survey, the 10 most important non routine skills for ICT SMEs managers and employees were: Communication, Team building, Planning, Learning and improvement, Openness to change, Creativity, Intelligibility, Decision Making, Innovativeness, Flexibility.

The consortium of the InTouch-ICT project developed 20 new games for mobile devices (2 for each non routine skill). The games were developed using the same methodology and the same technology of the previous ones, but new learning contents were created in order to match the specific learning needs of ICT SMEs.

III. THE SERIOUS GAMES

The games developed within the project take place in situations and contexts that are characteristic of day-to-day activities, namely within a small company titled "InTouch". The "InTouch" company is composed by several characters that were described giving their company role (Chief, PM-Design, PM-Development, PM-Assistance, Account, Account assistant, Supplier, Practitioner), personal information (name, surname, age, sex, star sign, hobbies), and a short bio (see Fig. 1).

Figure 1. Screenshots of the games. Clockwise from top left: the "InTouch" company team; a single character description (Chief); an interactive map example.

Games scenarios were obtained adapting situational cases found with the starting field research to the "InTouch" company and its characters.

The "InTouch" company elements that connect humor, sense and meaning, are characters' dossier and stories that were shortly given at the beginning of the games, and more extensively published on the "InTouch" Facebook page. Each dossier reported elements of the characters' lives, funny events from their past, additional information about their relationships, hobbies and funny photographs showing something weird about them. InTouch games, although short and simple, have thus a solid narrative structure, reinforced by the social media storytelling, in order to engage players, make them recognize narrative patterns referred to their work activities, and give them the right balance between fantasy and real working context situations.

In the development of the InTouch games attention was also paid to the ludic aspects. Even though challenges are not that complex, InTouch game design tried to respect requirements for the games to be relevant, explorative, emotive and engaging. Attention was paid to speed, level of difficulty, timing and range of feedback. Challenges of mastery and comprehension were inserted into games, together with strategy, and a perceived risk of failure to prevent boredom. Game mechanics were also made pleasant to create a positive climate, which is ideal when it comes to increase retention and recall. An entertaining gameplay was achieved through the use of funny graphics, novelty of the interactions, surprise, and humour in dialogues and scenarios.

All serious games were designed according to the same scheme, made of an opening scenario and a problem-based situation presenting the aim of the game (first frames), a bulk of interactivities (central frames) where players are asked to choose among different options, and the last frames showing the closing scenario, the score, and giving feedback to the player.

The narrative within the games is developed giving a short background story in the opening scenario, then it is influenced by user's action in the central frames, and ends up with the closing scenario.

The central frames are developed according to the following types of interaction:

- *Branching story*: the user reads the story and has to take different decisions. The story develops in different ways according to the choices made by users, and the final feedback and evaluation are the result of the combination of the choices.
- *Interactive map*: at the beginning of the game a problem-based situation is described in the "InTouch" company. To solve the problem, the user can choose three members of the company to talk to, but he/she needs to pick the right people to get the useful information. Once the user has read the three clues, he/she can choose one of the three available alternatives. Evaluation is based on the final decision and on the choice of the members of the "InTouch" company made by the player.
- *Multiple choice*: at the beginning of the game there is a description of a scenario and the aim of the game. The user has to help the main character with three different decisions in a limited time frame. The difficulty increases: in the first decision point only three out of the five listed options are correct, in the second one only two and in the third one only one. The final score and the feedback depend on how many correct answers the user chooses.
- *Quiz*: the game begins presenting a brief introduction of the main topic, then the player has to try to correctly answer three related questions. The player gets immediate feedback on the answer to each question and a summary at the end of game. The key objective is to gain points for fast and correct answers. Evaluation is based on a combination of the number of correct answers with the time taken to answer.
- *Task simulation*: during the game the user has to achieve a goal which foresees three different tasks. He/she has to make sure to do the right tasks in the right order, and then he/she has to answer to a question focused on the selected task. The score is determined from the number of correct answers and from the correctness of the order the user chose to prioritize the tasks.

IV. SCOPE AND HYPOTESES

This section illustrates what are the scopes of the research that was done on the developed serious games, and what are the hypothesis guiding it through the study.

The research can be divided in two steps, according to the different editions of the InTouch project.

In the first edition of the project (2010-2012) a kit of 30 serious games was tested, launching the study of the ludic and narrative components role within the games [1].

In the second edition (2013-2015) the structure of both ludic and narrative components was further analyzed studying a kit of 20 newly developed serious games.

On the first kit of 30 serious games a summative evaluation was conducted measuring a set of game variables on a sample of players. The four game variables of interest are: the players' willingness to play again, the interest of the goal, the fun of the gameplay, and the realism of the game narration.

Even if ludology and narratology are complex and multidimensional concepts, the fun of the gameplay and the realism of the game narration can be considered, at least partially, two components of these constructs, and their causal role within a serious game can shed light on the juxtaposition between ludology and narratology.

The interest of the goal is considered a primary element. It can be found starting from the beginning of the game, when the player faces the game scenario and mission.

It is then interesting to observe how the development of the game in terms of fun and narration can influence the causal relationship between the interest of the goal and the willingness to play again.

For this reason, the causal relationship among the interest for the goal and the willingness to play again is hypothesized to be mediated by the fun of the gameplay, and by the realism of the game narration.

The present study explores the degree to which the data fit different nested causal models. In the "complete" model (with less degrees of freedom), indicated as Model A (Fig. 2), the relationship between the interest of the goal and the willingness to play again is partially mediated both by the fun of the gameplay and by the realism of the game narration.

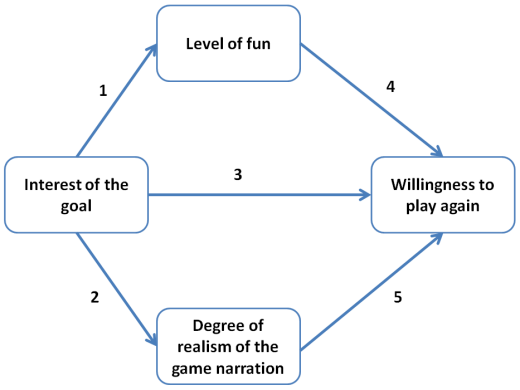


Figure 2. Graphical scheme of the causal Model A (first kit of serious games).

The fun of the gameplay and the realism of the game narration are hypothesized to positively influence the willingness to play again (paths 4 and 5). These hypotheses are based on the consideration that both the fun of the gameplay and the realism of the game narration are significant elements in determining the degree of satisfaction. It is also hypothesized that the interest of the goal positively influences the willingness to play again (path 3), since the engagement for the game mission can be considered as a natural predictor of the degree of satisfaction.

Some constraints of the complete model are then released, suppressing one or more causal paths, to obtain all the other nested models. In this way, the partial mediation of the fun of the gameplay and of the realism of the game narration will be substituted by their full mediation or by the lack of mediation. The complete Model A will thus be confronted with the following alternative models:

- Model B, where there is not mediation of the fun of the gameplay, path 1 is suppressed;
- Model C, where there is not mediation of the realism of the game narration, path 2 is suppressed;
- Model D, where there is not mediation either of the fun of the gameplay or of the realism of the game narration, paths 1 and 2 are suppressed;
- Model E, where there is full mediation both of the fun of the gameplay and of the realism of the game narration, path 3 is suppressed;
- Model F, where there is full mediation of the fun of the gameplay and there is not mediation of the realism of the game narration, paths 2 and 3 are suppressed;
- Model G, where there is not mediation of the fun of the gameplay and there is full mediation of the realism of the game narration, paths 1 and 3 are suppressed.

Table I summarizes which causal paths, indicated with numbers in Fig. 2, are present for each model.

TABLE I. CAUSAL PATHS OF TESTED MODELS

	Path 1	Path 2	Path 3	Path 4	Path 5
Model A	✓	✓	✓	✓	✓
Model B	--	✓	✓	✓	✓
Model C	✓	--	✓	✓	✓
Model D	--	--	✓	✓	✓
Model E	✓	✓	--	✓	✓
Model F	✓	--	--	✓	✓
Model G	--	✓	--	✓	✓

The comparison of nested models wants to establish if the hypothesized influence of the interest of the goal on the willingness to play again is better explained by partial mediation, full mediation, or no mediation at all of the fun of the gameplay and of the realism of the game narration.

Through the study on the second kit of 20 serious games the main scope of the research was to clarify if the ludic and the narrative components could be used to interpreter the causal structure of the games. For this reason, the set of considered variables was extended beyond those already analyzed during the first edition. The seven game variables of interest are in this case: the players' willingness to play again, the fun of the gameplay, the set of rules of the gameplay, the ways of interaction with the gameplay, the realism of the game narration, the narrative of the starting scenarios of the games, and the development of the narrative during the games.

The fun of the gameplay, the set of rules of the gameplay, and the ways of interaction with the gameplay are hypothesized to be indicators of the ludic factor of the games; while the realism of the game narration, the narrative of the starting scenarios of the games, and the development of the narrative during the games are considered as narrative indicators. This structure must be verified through a Confirmative Factor Analysis according to the scheme represented in Fig. 3.

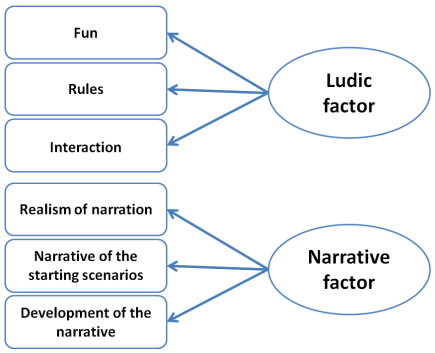


Figure 3. Graphical scheme of the factorial structure of the measured variables (second kit of serious games).

The willingness to play again is hypothesized to be positively influenced by all the other considered variables. As a consequence, the Structural Equation Model that will be tested can be represented with the ludic and narrative factorization, thus giving rise to the graphical scheme in Fig. 4, where paths are numbered starting from 6 to avoid confusion with the causal scheme referred to the first kit of serious games.

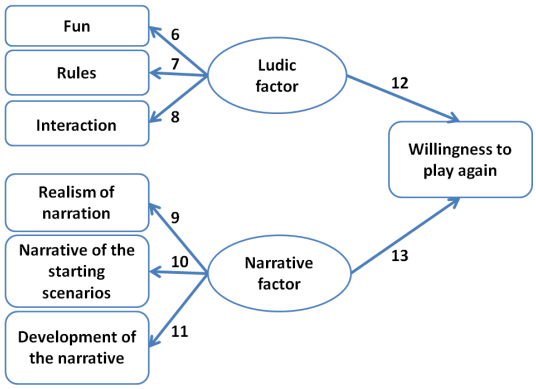


Figure 4. Graphical scheme of the Structural Equation Model (second kit of serious games).

V. METHODS

This section contains an illustration of the methods that were adopted in the present study: a description of the samples, the research procedure, the instruments, and the statistical analyses that were performed.

A. Participants

For the test of the first kit of 30 serious games the target sample consisted of 54 workers of 9 different SMEs from the eight countries participating in the first edition of the project and operating in different business sectors (ICT, business support, education/training, etc.). The SMEs were selected on the basis of their willingness to participate in the study. Work positions were: 28 managers and 26 employees. In total 30 were males (56%) and 24 were females (44%). The mean age was 41.94 years (SD = 9.70).

For the test of the second kit of 20 serious games the target sample consisted of 118 workers of 31 different SMEs from the five countries participating in the second edition of the project and operating mainly in the ICT sector. The SMEs were selected on the basis of their willingness to participate in the study. Work positions were: 23 managers and 95 employees. In total 61 were males (52%) and 57 were females (48%). The mean age was 36.64 years (SD = 7.35).

B. Procedure

To test the developed kits of mobile serious games the project partners held dedicated events, called "Learning Labs", in each country participating in the project for both the editions of the project. During each Learning Lab a structured questionnaire, with only slight differences in the two editions, was proposed to participants after the completion of the games. Participation to the Learning Labs and questionnaire compilation were obtained through an informed consent procedure asking for active consent from participants. Questionnaires took approximately 30 minutes to complete. Project staff members introduced the questionnaires, giving instructions about their compilation, explaining that they were voluntary and responses were anonymous and confidential. Project staff members were at the workers' disposal during the questionnaires' administration to answer questions and give explanations.

C. Measures

An Identifying Information Form was used to collect demographic information: age, gender, working role.

An articulated grading grid was proposed to participants, after the completion of the games, asking them to express on a 10 point Likert scale their like about several game variables.

The present study is taking in consideration the following variables: the willingness to play again ("Would you like to play again?"), the fun of the gameplay ("How fun was your interaction with the game mechanics?"), the interest of the goal ("How interesting was the goal proposed by the game?"), the realism of the game narration ("If compared to your experience, how realistic was the narrative of the game about the 'InTouch' company?"), the set of rules of the games ("How did you like the rules of the games?"), the ways of interaction with the gameplay ("Rate your satisfaction for the interactions within the games"), the narrative of the starting scenarios of the games ("How did you like the narrative of the scenarios that are proposed at the beginning of the games?"), and the development of the narrative during the games ("How did you like the evolution of the narrative during the games?").

D. Data Analysis

Preliminary Analysis

As a preliminary analysis, skewness and kurtosis of all game variables were checked. Overall, all variables showed to conform to the normal distribution.

Correlation

As a first step, the correlation matrix of both the set of variables measured by the questionnaires was calculated.

1) First kit of 30 serious games

For the first kit of 30 serious games a path analysis was done to test the casual models represented in Fig. 2 and to establish which model was to be preferred. All path models involving the aforementioned variables were analyzed with LISREL, using maximum likelihood estimation procedures [23]. For each tested model χ^2 is reported, as an absolute fit index (good fit between zero value and two times the degrees of freedom). Three more fit indexes were also reported: the non-normed fit index (NNFI); the comparative fit index (CFI); and the root mean square error of approximation (RMSEA). Higher CFI and NNFI values (in the range from 0.97 to 1.00 for a good fit) and lower

RMSEA values (in the range from 0.00 to 0.05 for a good fit) are assumed to evaluate model fit [24]. The Coefficient of determination (R-square) is reported, giving the percentage of variance of the willingness to play again explained by each model, to estimate the completeness of the considered set of predictors.

To establish which type of mediation (partial, full, or non-significant) was exercised by the fun of the gameplay and by the realism of the game narration, the comparison of the fit of alternative nested models was conducted analyzing for each pair of models the differences of the χ^2 values (indicated with $\Delta \chi^2$) between the less parsimonious model (i.e., the one with less degrees of freedom, in our case the complete Model A) and the more parsimonious one (i.e., in turn: Models B, C, D, E, F, and G). The significance of $\Delta \chi^2$ has successively been established looking at the p-value corresponding to the χ^2 distribution for a number of degrees of freedom given by the difference of degrees of freedom of the more parsimonious models and the complete one. Choosing a cut-off of $p = 0.01$, if the $\Delta \chi^2$ between two nested models is significant ($p < 0.01$), this implies that the complete model explains the data better; if there is no significant difference between two nested models ($p > 0.01$), this implies that the more parsimonious model explains the data equally well compared to the complete model, and must be preferred for its simplicity.

2) Second kit of 20 serious games

For the second kit of 20 serious games a preliminary Confirmatory Factor Analysis (CFA) was done to verify the correctness of the factorization with the ludic and the narrative factors, as represented in Fig. 3. After that, a complete Structural Equation Model, made of both measured and latent variables, as represented in Fig. 4, was analyzed with LISREL [23], reporting the same fit indexes already used with the first kit of games.

Thanks to the large enough sample size for the testing of the second kit of 20 serious games, it was possible to estimate the numerical values of the causal paths for different groups of respondents. For the reliability and validity of results it was chosen to differentiate males (61) from females (57), and the younger than 35 years (58) from the older than 35 years (60) people. For these categories of respondents, in fact, each group was enough large. On the contrary, the scarcity of managers (23) in the considered sample prevented to confront them with employees (95), since the results were at risk to be neither reliable or valid.

The multi-group analysis was conducted through a hierarchical increasingly restrictive set of steps [25]. It started with the determination of a well-fitting multi-group baseline model with the same pattern of parameters across groups (called "Configural model"). The subsequent tests involved the specification of cross-groups equality constraints for measurement equivalence, followed by structural equivalence. The comparison of the baseline model with the measurement invariant model, and of this latter with the structural invariant model, was done in the same way as for the nested models already exposed for the first kit of serious games. For each step (baseline - measurement invariance - structural invariance) the number of degrees of freedom grows, as an effect of the added invariance constraints, and the value of χ^2 for the fit is likely to grow. If the growth of χ^2 is not significant ($p > 0.01$) when considering a nested model, this latter must be preferred, and the invariance is demonstrated.

For the comparison of two groups, at least measurement invariance must be demonstrated, meaning that the form of the causal model is the same for both groups. After that, the similarity of parameters' values within the common form is studied trough the analysis of structural invariance across groups.

For the considered sample it was tested if the same form was valid for males and females, and for younger and older respondents. If so, the comparison of structural parameters was done, establishing if differences in causal paths' values were significant.

VI. RESULTS

This section contains the numerical results obtained for the previously illustrated data analysis: correlation, path analysis, comparison of nested causal models, Confirmatory Factor Analysis, Structural Equation Model, and multi-group analysis.

1) First kit of 30 serious games

Table II reports the correlation coefficients of the willingness to play again, the fun of the gameplay, the interest of the goal, and the realism of the game narration calculated with the first kit of 30 serious games. The level of significance (p-value) is indicated in the table footnote.

TABLE II. CORRELATION MATRIX OF THE FIRST KIT OF 30 SERIOUS GAMES (SUB-DIAGONAL COEFFICIENTS).

Variable	Willingness to play again	Level of fun	Interest of the goal
Level of fun	0.89*		
Interest of the goal	0.60*	0.35*	
Realism of the game narration	0.21	-0.12	0.19

b. *p < 0.01.

Table III reports the results of the path analysis for the seven tested models with the levels of significance of the causal paths (p-values) indicated in the table footnote. Path numbers are those indicated in Fig. 2.

TABLE III. PATH ANALYSIS COEFFICIENTS FOR THE TESTED MODELS

	Path 1	Path 2	Path 3	Path 4	Path 5
Model A	0.35*	0.19	0.26**	0.83*	0.26**
Model B	--	0.26**	0.26**	0.83*	0.26**
Model C	0.39*	--	0.26**	0.83*	0.26**
Model D	--	--	0.26**	0.83*	0.26**
Model E	0.35*	0.19	--	0.93*	0.32**
Model F	0.39*	--	--	0.93*	0.32**
Model G	--	0.26**	--	0.93*	0.32**

c. * p < 0.01; ** p < 0.05.

Table IV reports the results of the comparison of the fit of the seven tested models, whose degrees of freedom (df) are indicated in the table, and with the level of significance of the difference between the complete and the nested models indicated in the table footnote.

TABLE IV. COMPARISON OF THE FIT OF ALTERNATIVE NESTED MODELS

Model	χ^2	NNFI	CFI	RMSEA	R^2	df	
A	0.00	1.00	1.00	0.00	0.95	1	
Alternative nested models							$\Delta\chi^2$
B	6.49	0.91	0.97	0.21	0.94	2	6.49
C	1.91	1.00	1.00	0.00	0.95	2	1.91*
D	9.39	0.91	0.96	0.20	0.94	3	9.39
E	27.79	0.31	0.77	0.50	0.89	2	27.79
F	28.85	0.53	0.77	0.41	0.89	3	28.85
G	31.09	0.47	0.73	0.42	0.89	3	31.09

d. *p > 0.01; R^2 = explained variance of the willingness to play again.

Looking at the results of the comparison of the nested models, Model C explains the data equally well compared to the complete Model A (p > 0.01) and must be preferred, being more parsimonious.

For the selected Model C the effects of the three predicting variables (Interest for the goal, Fun of the gameplay, Realism of the game narration) on the Willingness to play again (outcome variable) were calculated and are reported in Table V.

TABLE V. EFFECTS ON THE WILLINGNESS TO PLAY AGAIN (MODEL C)

Variable	Total effect	Direct effect	Indirect effect
Level of fun	0.83*	0.83*	--
Interest of the goal	0.58*	0.26**	0.32**
Degree of realism of the game narration	0.26**	0.26**	--

e. * p < 0.01; ** p < 0.05

Both the fun of the gameplay and the realism of the game narration have significant direct effects on the willingness to play again (path 4 = 0.83; path 5 = 0.26).

The interest of the goal has a significant total effect on the willingness to play again, obtained as the sum of a direct effect (path 3 = 0.26) and an indirect effect (path 1 × path 4 = 0.32) through the mediation of the fun of the gameplay.

The study on the first kit of 30 serious games demonstrated that, as hypothesized, both the fun of the gameplay and the realism of the game narration significantly influence the willingness to play again. Causal paths 4 and 5, in fact, are significant across all tested models.

In particular, the influence of the fun of the gameplay resulted to be more robust, with values of path 4 above 0.80, while the influence of the realism of the game narration, even though significant, was less pronounced, with values of path 5 around 0.30.

Furthermore, the fun of the gameplay resulted to significantly mediate the relationship between the interest of the goal and the willingness to play again. On the contrary, no significant mediation emerged for the realism of the game narration, insomuch as the causal Model C, where path 2 is suppressed, was preferred.

As a whole, the relationship between the interest of the goal and the willingness to play again is partially mediated by the fun of the gameplay, and non-significantly mediated by the realism of the game narration.

2) Second kit of 20 serious games

Table VI reports the correlation coefficients of (a) the willingness to play again, (b) the fun of the gameplay, (c) the set of rules of the games, (d) the ways of interaction with the gameplay, (e) the realism of the game narration (f), the narrative of the starting scenarios of the games, and (g) the development of the narrative during the games calculated with the second kit of 20 serious games.

TABLE VI. CORRELATION MATRIX OF THE SECOND KIT OF 20 SERIOUS GAMES (SUB-DIAGONAL COEFFICIENTS)

Variable	a	b	c	d	e	f
b	0.58*					
c	0.68*	0.76*				
d	0.51*	0.79*	0.70*			
e	0.32*	0.14	0.11	0.17		
f	0.33*	0.18	0.02	0.11	0.69*	
g	0.30*	0.20**	0.11	0.18	0.59*	0.79*

f. * p < 0.01; ** p < 0.05

Table VII reports the result of the Confirmatory Factor Analysis with the introduction of the ludic and narrative factors, as represented in Fig. 3.

TABLE VII. CONFIRMATORY FACTOR ANALYSIS LOADINGS

Variable/Factor		Ludic factor		Narrative factor
Level of fun		0.93*		--
Rules		0.81*		--
Interaction		0.85*		--
Degree of realism of the game narration		--		0.72*
Narrative of the starting scenarios		--		0.95*
Development of narrative		--		0.83*
χ^2	**NNFI**	**CFI**	**RMSEA**	**df**
14.92	0.97	0.98	0.09	8

g. * p < 0.01

The introduction of the ludic and narrative latent factors shows a good fit with the data, and the loading factors are high for all the considered variables.

This result encourages to further continue in the use of the ludic and narrative latent factors to study the Structural Equation Model for the prediction of the willingness to play again.

Table VIII reports the results of the Structural Equation Model with paths numbered as in Fig. 4.

TABLE VIII. STRUCTURAL EQUATION MODEL COEFFICIENTS

Path	6	7	8	9	10	11	12	13
Coeff.	0.89*	0.86*	0.84*	0.72*	0.96*	0.82*	0.65*	0.26**
Fit index		χ^2		NNFI	CFI	RMSEA	df	
		40.91		0.91	0.95	0.13	13	

h. *p < 0.01, ** p < 0.05

The Structural Equation Model shows an overall good fit with the data. As a consequence, the casual model for the prediction of the willingness to play again can be adequately realized with the use of the two latent factors referred to the ludic and narrative components of the games. With such a schematization, both the ludic and the narrative contribute to the willingness to play again resulted to be significant, even if the ludic component (path 12) showed to be stronger than the narrative one (path 13).

The introduction of latent factors representing ludic and narrative components did not significantly worsen the fit with data if compared with the path analysis conducted with the first kit of serious games, where only measured variables were used. This is an important result that confirms the legitimacy of the interpretation of the casual relationships in terms of ludic/narrative contributes. Furthermore, the second study substantially confirms the same distribution of weight between ludic and narrative contribution to the willingness to play again that was found with the first study. Also, this result is encouraging, since it confirms that similar findings were obtained with different samples of users, thus cross-validating the ludic/narrative key of lecture that was adopted for both studies.

The results of the multi-group analysis, referred to males vs. females, and to younger than 35 years vs. older than 35 years, are reported in Table IX, where the baseline model (same patterns) is compared through the χ^2 distribution to the measurement invariant model, and this latter is compared to the structural invariant model.

TABLE IX. MULTI-GROUP ANALYSIS RESULTS

Groups	Configural Model df = 26	Measurement Invariant Model df = 40	Structural Invariant Model df = 42
Males (61) vs. Females (57)	$\chi^2 = 35.14$	$\chi^2 = 40.94$ $p\ (\Delta\chi^2)_{14} = 0.97$	$\chi^2 = 50.88$ $p\ (\Delta\chi^2)_2 < 0.01$
Younger (58) vs. Older (60)	$\chi^2 = 43.60$	$\chi^2 = 48.08$ $p\ (\Delta\chi^2)_{14} = 0.99$	$\chi^2 = 58.72$ $p\ (\Delta\chi^2)_2 < 0.01$

For both the pairs of groups the measurement invariance is verified, while there is a significant structural difference. In practice, measurement invariance corresponds to the similarity for each separated group of the factorialization with the Ludic and the Narrative factors that was done in the analyzed models. That is, independently from which group is being considered (males or females, younger or older people), the fun of the gameplay, the set of rules of the games, and the ways of interaction with the gameplay contribute similarly to the ludic factor; while the realism of the game narration, the narrative of the starting scenarios of the games, and the development of the narrative during the games contribute similarly to the narrative factor. In terms of Structural Equation Model, it is expected that coefficients of paths 6, 7, 8, 9, 10, and 11, as numbered in Fig. 4, are similar across groups.

On the contrary, each pair of groups has significant structural differences across groups, meaning that causal paths coefficients of the structural part of the models (paths numbered 12 and 13) are significantly different for males and females, and for younger and older respondents. In practice, the ludic and the narrative contribution to the willingness to play again strictly depends on what group is being considered.

Table X reports the results of the Structural Equation Model for separated groups of males and females with paths numbered as in Fig. 4.

TABLE X. STRUCTURAL EQUATION MODEL COEFFICIENTS (MALES VS. FEMALES)

Path	6	7	8	9	10	11	12	13
Males (61)	0.88*	0.84*	0.83*	0.70*	0.94*	0.81*	0.59*	0.26**
Females (57)	0.88*	0.83*	0.85*	0.82*	0.85*	0.77*	0.32*	0.76*

i. *p < 0.01, ** p < 0.05

As anticipated by the multi-group analysis, there is a strong similarity among those coefficients (numbered from 6 to 11) that refer to the ludic and narrative factorialization; while significant differences are present for coefficients 12 and 13, which correspond to the ludic and narrative contribution to the willingness to play again. Both for males and females the willingness to play again is significantly influenced by ludic and narrative components, but their order of importance is inverted. Males are mainly influenced by the ludic factor, while females are mainly influenced by the narrative factor

Table XI reports the results of the Structural Equation Model for separated groups of younger than 35 years and older than 35 years, with paths numbered as in Fig. 4.

TABLE XI. STRUCTURAL EQUATION MODEL COEFFICIENTS (YOUNGER THAN 35 YEARS VS. OLDER THAN 35 YEARS)

Path	6	7	8	9	10	11	12	13
Younger	0.90*	0.85*	0.85*	0.72*	0.96*	0.82*	0.61*	0.25**
Older	0.91*	0.83*	0.86*	0.81*	0.89*	0.81*	0.34*	0.72*

j. *p < 0.01, ** p < 0.05

As anticipated by the multi-group analysis, and already seen in the case of males vs. females, also, for the differentiation between younger and older people of the sample, there is a strong similarity among coefficients numbered from 6 to 11; while significant differences are present for coefficients 12 and 13. Even if both the ludic and

the narrative factor contribute significantly to the willingness to play again, younger people of the sample are mainly influenced by the ludic factor, while older ones are mainly influenced by the narrative factor.

VII. Conclusion and Future Work

A. Contribution to the ludology/narratology debate

Interpreting the fun of the gameplay as a ludic indicator, and the realism of the game narration as a narrative indicator, the results of the study conducted on the first set of 30 serious games can be referred to the ludology/narratology debate.

The results seem to corroborate a point of view that takes in consideration both positions. This sort of reconciliation of the two different positions, however, is not gained through an assimilation of the realism of the game narration to the fun of the gameplay. As reported in Table II, in fact, their correlation coefficient is non-significant (and slightly negative), indicating their substantial independence (or even slight juxtaposition). The fun of the gameplay and the realism of the game narration must therefore be considered as separately, and differently, contributing to determine the success of a learning game.

The results of the study on the first kit of 30 serious games seem to mostly corroborate Jenkins' proposal of "game space", whose structure facilitates narrative experience [20]. In this sense, the interest of the goal can be interpreted as a feature of the "game space" that can enhance the degree of satisfaction of the players, determining their retention in a direct way, and indirectly, thanks to its contribution to the fun of the gameplay.

B. Limits and specificity of the study

If limited to the study on the first kit of 30 serious games, the association of the fun of the gameplay and of the realism of the game narration with the ludic and the narrative components of a serious games is exposed to criticism of being both partial and spurious. While the significance of the results is robustly consistent with the measured variables, it must be recognized that different types of narrative can be developed within a serious game, not limited to realistic ones. Having considered the realism of the narration is certainly only a partial representation of the narrative of a serious game. At the same time, fun in a serious narrative game can derive not only from the act of playing, but also from other components, like the fact to learn something interesting, or to take part in an engaging story. The fun of the gameplay can thus be referred not only, or at least not exclusively, to the ludic aspects of a serious game.

To have a more comprehensive insight of the ludic and narrative dynamics within a serious game, a larger number of indicators should be analyzed and validated, as referred to the ludic and to the narrative constructs. One more limit of the study conducted on the first kit of 30 serious games is the small size of the sample (n = 54) that prevented, for instance, to study differences between groups of participants for the scarcity of people in each group.

Some of the limits of the first study were addressed with the testing of the second kit of 20 serious games. The size of the sample was doubled (n=118), and the measured variables were explicitly treated as indicators of the ludic and narrative factors.

The prevalence of the ludic contribute to the willingness to play again was found in both the studies that were presented in this paper. This result, however, must be considered as strictly linked to the specific serious games that were analyzed and cannot be generalized to every game-based solution. Furthermore, it was found that when differentiating groups within the sample, for instance according to gender and age, the prevalence of the ludic and/or of the narrative factor can change. In the analyzed sample, males and younger people showed to be more influenced by the ludic component, while females and older people resulted to be more influenced by the narrative component.

The tested games were very basic as a consequence of their design for mobile devices. The serious games were conceived to be played anywhere and at any time, taking few minutes to be completed. Games' interface was designed in such a way that a simple touch, or click, was enough to interact, thus enabling one-hand playing. This simplicity can be the origin of the ludic predominance over the narrative. It can be hypothesized that game-based solutions with a predominant ludic component exhibit a behavior like the one found in this case. In different situations, for those game-based solutions that are more focused on the narrative component, or where both the ludic and the narrative components are equally juxtaposed, different solutions can be found.

C. Perspectives of implementation

The differences across groups (males vs. females, and younger vs. older people) that were found on the considered sample deserve to be more deeply studied and related to the existing literature in the field of game studies. Beside gender and age, it is worth analyzing the role of the ludic and of the narrative components according to the work position (employers vs. employees). This kind of analysis was not possible in the present study because the sample was not adequate for the scope (there were too few managers in the considered sample). In general, multi-group analysis was not among the initial objectives of the present studies, while gender differences and other issues that can be referred to game studies must be part of the experimental design from the beginning of the study to be adequately treated. All these suggestions must be verified and constitutes an indication for future work in view of an implementation of the proposed research concept with other serious games.

An interesting perspective of implementation comes from the substantial independence that was found between the ludic and the narrative contributes within the games. In both the presented studies, in fact, the correlation between the ludic and the narrative components resulted to be non-

significant. This suggests that, when designing a serious game, ludic and narrative components must be considered as separately, and differently, contributing to determine its overall success. Under an operational point of view, it would be a significant added value the creation of a validated instrument measuring ludic and narrative components within a serious game through the use of exploratory factorial analysis with multiple items, instead of a self-developed questionnaire, with one item for each variable, like the one that was used for the present study with the weak validation of the confirmatory factor analysis that was shown above.

ACKNOWLEDGMENT

The InTouch Project was funded by the European Commission within the Lifelong Learning "Leonardo da Vinci" Multilateral projects Programme. The first edition (2010-2012) was a Development of Innovation project, the second edition, namely InTouch-ICT (2013-2015), was a Transfer of Innovation project. InTouch success was due to the support and commitment of all partners: Centre for Flexible Learning - CFL, municipality of Soderhamn (Sweden); Enocta (Turkey); Faculty of Economics and Management, Kaunas University of Technology (Lithuania); Exemplas Holdings Limited (United Kingdom); Centro per le Applicazioni della Televisione e delle Tecniche di Istruzione a Distanza - CATTID, University of Rome "La Sapienza" (Italy); CIBC Artois Ternois (France); Evolaris next level GmbH (Austria); Centro Italiano per l'Apprendimento Permanente – CIAPE (Italy); Bulgarian Development Agency (Bulgaria); The Swiss Federation for Adult Learning - SVEB (Switzerland); Sapienza Innovazione (Italy); Okan University (Turkey); Refile (Italy); Trebag (Hungary); TBV (Turkey).

REFERENCES

[1] A. Imbellone, B. Botte, and C.M. Medaglia, "An Empirical Study on the Ludic and Narrative Components in Mobile Game-Based Learning," Proc. The Seventh International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2015), IARIA, Lisbon, 22-27 February 2015, pp. 8-13, ISSN: 2308-4367, ISBN: 978-1-61208-385-8.

[2] A. Amory, "Game object model version II: a theoretical framework for educational game development," Education Tech Research Dev, vol. 55, Issue 1, pp. 51–77, February 2007.

[3] M. Sharples, "Mobile learning: research, practice and challenges," Distance Education in China, vol. 3, Issue 5, pp. 5-11, March 2013.

[4] D. Charsky, "From edutainment to serious games: A change in the use of game characteristics," Games and Culture: A Journal of Interactive Media, vol. 5, Issue 2, pp. 177-198, April 2010.

[5] C. Girard, J. Ecalle, and A. Magnan, "Serious games as new educational tools: How effective are they? A meta-analysis of recent studies," Journal of Computer Assisted Learning, vol. 29, Issue 3, pp. 207-219, June 2013.

[6] S. de Freitas and H. Routledge, "Designing leadership and soft skills in educational games: The e-leadership and soft skills educational games design model (ELESS)," British Journal of Educational Technology, vol. 44, Issue 6, pp. 951-968, November 2013.

[7] P. Wouters, C.van Nimwegen, H. van Oostendorp, and E.D. van der Spek, "A meta-analysis of the cognitive and motivational effects of serious games," Journal of Educational Psychology, vol. 105, Issue 2, pp. 249-265, May 2013.

[8] I. Mayer et al., "The research and evaluation of serious games: Toward a comprehensive methodology," British Journal of Educational Technology, vol. 45, Issue 3, pp. 502-527, May 2014.

[9] E.A. Boyle, T.M. Connolly, and T. Hainey, "The role of psychology in understanding the impact of computer games," Entertainment Computing, vol. 2, Issue 2, pp. 69–74, January 2011.

[10] J.H. Brockmyer et al., "The development of the Game Engagement Questionnaire: a measure of engagement in video game-playing," Journal of Experimental Social Psychology, vol. 45, Issue 4, pp. 624–634, March 2009.

[11] D.K. Mayes and J.E. Cotton, "Measuring engagement in video games: A questionnaire," Proc. Human Factors and Ergonomics Society Annual Meeting, SAGE Publications, vol. 45, Issue 7, pp. 692-696, Minneapolis, 8-12 October 2001.

[12] T.M. Connolly, M. Stansfield, and L. Boyle (Eds.), "Games-Based Learning Advancements for Multi-Sensory Human Computer Interfaces: Techniques and Effective Practices," Hershey, PA: Information Science Reference, 2009.

[13] T. Hainey and T.M. Connolly, "Evaluating games-based learning," International Journal of Virtual and Personal Learning Environments, vol. 1, Issue 1, pp. 57–71, March 2010.

[14] J. Juul, "Games telling stories? A brief note on games and narratives," Game studies: The International Journal of Computer Game research, vol. 1, Issue 1, July 2001.

[15] G. Frasca, "Simulation versus Narrative: introduction to ludology," in The video game theory reader, M. J. P. Wolf and B. Perron, (Eds.) New York: Routledge, pp. 221-236, 2003.

[16] A. McManus and A.H. Feinstein, "Narratology and ludology: competing paradigms or complementary theories in simulation," Developments in Business Simulation and Experiential Learning, vol. 33, pp. 363-372, March 2006.

[17] M.L. Ryan, "Beyond Myth and Metaphor: The Case of Narrative in Digital Media," Game studies: The International Journal of Computer Game research, vol. 1, Issue 1, July 2001.

[18] E.J. Aarseth, "Cybertext. Perspectives on Ergodic Literature," Baltimore, MD: Johns Hopkins University Press, 1997.

[19] C.A. Lindley, "Ludic Engagement and Immersion as a Generic Paradigm for Human-Computer Interaction Design," Proc. The Third International Conference on Entertainment Computing (ICEC 2004), September 2004, pp. 3-13, ISBN: 978-3-540-22947-6.

[20] H. Jenkins, "Game design as narrative architecture," in First person: New media as story, performance, and game, N. Wardrip-Fruin and P. Harrigan (Eds.), Cambridge, MA: MIT, pp. 118-130, 2004.

[21] T.M. Connolly, E.A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of the empirical evidence on computer games and serious games," Computers & Education, vol. 59, Issue 2, pp. 661–686, September 2012.

[22] J. Froschauer, J. Zweng, D. Merkl, M. Arends, D. Goldfarb, and M. Weingartner, "ARTournament: A Mobile Casual Game to Explore Art History," Proc. 12th IEEE International Conference on Advanced Learning Technologies (ICALT 2012), IEEE Press, pp. 80-84, July 2012, ISBN: 978-1-4673-1642-2.

[23] K.G. Joreskog and D. Sorbom, LISREL 8.8 for Windows [Computer software]. Skokie, IL: Scientific Software International, Inc., 2006.

[24] K. Schermelleh-Engel, H. Moosbrugger, and H. Muller, "Evaluating the Fit of Structural Equation Models: Tests of Significance and Descriptive Goodness-of-Fit Measures," Methods of Psychological Research-Online, vol. 8, Issue 2, pp. 23-74, 2003.

[25] K. A. Bollen, "Structural equations with latent variables," Hoboken, NJ: John Wiley & Sons, 2014.

On Multi-controller Placement Optimization in Software Defined Networking - based WANs

Eugen Borcoci, Tudor Ambarus, Marius Vochin
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
eugen.borcoci@elcom.pub.ro, tudorambarus@yahoo.com, mvochin@elcom.pub.ro

***Abstract* — Software Defined Networking (SDN) is a recent networking technology that promises important advantages in IP networking, related to flexibility at network and application level, together with powerful management and control. However, the SDN specific centralization creates scalability problems in large network environments. Multi-controller implementation of the SDN control plane for large networks can solve the scalability and reliability issues introduced by the SDN centralized logical control principle. There are still open research issues related to controllers placement, static or dynamic assignment of the network forwarding nodes to controllers, especially when network nodes/links and/or controllers failures appear or some constraints are imposed. This paper contains an analysis of some solutions proposed in the literature followed by a work in progress, on multi-criteria optimization methods applicable to the controller placement problem.**

Keywords — Software Defined Networking; Distributed Control Plane; Controller placement; Reliability; Multi-criteria optimizations.

I. INTRODUCTION

The recently proposed Software Defined Networking (SDN) technology offers significant advantages to cloud data centers and also to Service Provider Wide Area Networks (WAN)[1]. The *basic principles* of the SDN architecture are [2][3][4]: *clear decoupling of the control and forwarding* (data) planes; *logically centralized control*; exposure of *abstract vision on network resources and state to external applications*. Thus, SDN offers an important advantage of independency of the control software w.r.t. forwarding boxes implementations offered by different vendors. Higher degree of programmability of the network control and also of applications is important consequences of the above principles.

This paper considers the case when SDN–type of control is applied in a WAN, owned by an operator and/or a Service Provider (SP).

Open research issues are related to the fact that the control-data plane separation can generate performance limitations and also reliability issues of the SDN controlled network [5][6] (note that in the subsequent text, by "controller" it is understood a geographically distinct controller location):

(a) The forwarder nodes (called subsequently "forwarders" or simply "nodes") must be continuously controlled, in a proactive or reactive way. The forwarders have to ask their master controllers and then be instructed by the latter, on how to process various new flows arriving to the forwarders. The result of instructions issued by the controller is filling appropriately the *flow tables* in the forwarder [2]. Such tables show what sequence of processing actions should be applied to a given data flow (identified by some specific parameters). The control communication overhead (and its inherent delay), between several forwarders and a single controller, can significantly increase the response time of the overall system. This happens because any type of controller would have eventually a limited processing capacity [5], w.r.t. the number of flow-related queries, or equivalently, the number of forwarders assigned to a controller could be too high. Therefore, the problem is: *how to distribute the controllers* as to minimize the controller-forwarder delay for a given network?

(b) The SDN control plane computes a single logical view upon the network; to this aim *the controllers must inter-communicate* and update/synchronize their data bases, in order to support the logical view construction and continuously updating of this unique vision upon the network [7][8][9]. One possible solution for inter-controller communication is to create an overlay network linking the controllers on top of the same infrastructure used by the data plane flows [10]. The problem is: how to distribute the controllers as to minimize the controller-to-controller communication delay, for a given set of controllers?

(c) Asynchronous events, such as controller failures or network disconnections between the control and data planes, may also lead to packet loss and performance degradation [5][11]. One can suppose that some forwarders are still alive (i.e., they can continue to forward the traffic flows, conforming their current flow table content). However, if the forwarders can no longer communicate with some controller, then they will have no knowledge on how to process the newly (in the future)

arrived flows. The problem is: how to distribute the controllers as to minimize the controllers and/or network links failures consequences?

There is a need to optimally place the controllers in the network. The overall objective will be an attempt to solve as much as possible of (a), (b), and (c) problems. However, this is a multi-criteria optimization problem and it was recognized as being a NP-hard one [11]. Consequently, different solutions have been proposed, targeting performance (problem (a), (b)), and performance plus reliability (problem (c)).

The paper [1] presented at The International Symposium on Advances in Software Defined Networks SOFTNETWORKING 2015, in Barcelona, Spain, contained an analysis of some solutions for (a), (b), (c) and then proposed a preliminary contribution on how *multi-criteria optimization algorithms* can be applicable to the controller placement problem. The study target was *not to develop specific algorithms*, dedicated to find an optimum solution for *a single given criterion* (several studies did that), but to *achieve an overall optimization of the controller placement*, by applying *multi-criteria decision algorithms (MCDA)* [12] [13]. The input data of the MCDA is a set of candidates (here an instance of controller placement is called a *candidate solution*).

This work is an extension of the paper [1], by refining more deeply some subjects and also by adding a novel section dedicated to simulation experiments and results.

The material is organized as follows. Section II is an overview of related work. Section III outlines several metrics and algorithms used in optimizations and present some of their limitations. Section IV develops the framework for MCDA usage as a tool for final selection of the control placement solution. Section V (which is new in this paper w.r.t. [1]) presents simulation model and a set of simulation experiments performed by the authors and results. Section VI presents conclusions and outlines the future work.

II. Related Work on SDN Controller Placement

This section is a short overview on some previously published work on controller placement in SDN - managed WANs. It is supposed that network topology and some metrics are known. The basic problems to be solved (in an optimum way) are: (1) *how many SDN controllers are needed* for that network and (2) *where the controllers should be placed* in the network. The goal is to provide enough performance (e.g., low delay for controller-to-forwarder communications) and also robustly preserve the performance level, when some controllers and/or network failures occur. Intuitively, it can be seen that some trade-off will be necessary.

In the cases of WANs having significant path delays, the controller placement determines the control plane convergence time. In other words, the placement affects the controller's response to real-time events sensed by the forwarders, or, in case of proactive controller-initiated actions, how fast the controllers can push (in advance) the required actions to forwarding nodes.

Actually, it has been shown in [10][14] that such a problem is theoretically not new. If *latency* is taken as a metric, the problem is similar to the known one, as *facility or warehouse location problem*, solved, e.g., by using Mixed Integer Linear Program (MILP) tools.

The Heller et al. early work [11] motivates the controller placement problem and then quantifies the placement impact on real topologies like Internet2 [5] and different cases taken from Internet Topology Zoo [16]. Actually, their main goal was not to find optimal minimum-latency placements (they observed that generally, such a problem has been previously solved) – but to present an initial analysis of a fundamental design problem, still open for further study. It has been shown that it is possible to find optimal solutions for realistic network instances, in failure-free scenarios, by analyzing the entire solution space, with off-line computations. This work also emphasized the fact (apparently surprising) that in most topologies, one single controller is enough to fulfill existing reaction-time requirements for some, reasonable size networks. However, resiliency aspects have not been considered in the above study.

Several works [10][14][17][18][19] have shown that resilience is important in the context of SDN and especially in the cases where additionally Network Function Virtualization (NFV) is wanted. Some resiliency-related issues have been considered in [14]:

(1) *Controller failures*: in case of a primary controller failure, it should be possible to reassign all its previously controlled nodes to a secondary closest controller, by using a backup assignment or via signaling, based on normal shortest path routing. Extreme case scenarios have been also considered, e.g., if at least one controller is still reachable; in such a case all nodes should keep functioning by communicating with it.

(2) *Network Disruption*: the failure of network links/ nodes may appear, altering the topology. The routing paths (and their associated latencies) will change; a novel reassignment of nodes (forwarders) to other reachable controllers is needed. In the worst case, some parts of the network can be completely cut off, having no access to controllers. On short term this problem has no solution: such separated nodes can still forward traffic based on existing flow tables content, but they would have no more a controller to which send their request and from whom receive new instructions.

(3) *Controller overload* (load imbalance): shortest path–based assignment of the forwarders to controllers is natural. However, one should avoid that one controller have too many nodes to manage; otherwise its average response time will increase. Therefore, a well-balanced assignment of nodes to the different controllers is needed.

(4) *Inter-Controller Latency*: the SDN concepts ask for a centralized logic view of the network; therefore, inter—controller communications are necessary to synchronize

their data bases. No matter if a single flat level of controllers (e.g., like in Onix [8]) or a hierarchical topology (e.g., like in Kandoo [9]) of controllers is used, it is clear that inter-controller latency should be minimized. Therefore, an optimized controller placement should meet such a requirement.

The works [10][18] present a metric to characterize the reliability of SDN control networks. Several placement algorithms are developed and applied to some real topologies, claiming to improve the reliability of SDN control, but still keep acceptable latencies. The controller instances are chosen such that the chance of connectivity loss is minimized; connections are defined according to the shortest path between controllers and forwarding devices.

The work [19] identifies several limitations of previous studies:

(1) forwarder-to-controller connectivity is modeled using single paths; yet, in practice, multiple concurrent connections may be available;

(2) peaks in the arrival of new flows are considered to be only handled *on-demand*, assuming that the network itself can sustain high request rates;

(3) failover mechanisms require predefined information which, in turn, has been overlooked.

The paper proposes the *Survivor*, i.e., a controller placement strategy that explicitly considers for network design the following elements: path diversity, controller capacity awareness, and failover mechanisms. Specific contributions consist in: significant reduction of the connectivity loss by exploring the path diversity (i.e., connectivity-awareness), which is shown to reduce the probability of connectivity loss in around 66% for single link failures; considering capacity-awareness proactively, while previous work handled requests churn on demand (it is shown that capacity planning is essential to avoid controller overload, especially during failover); smarter recovery mechanisms by proposing heuristics for defining a list of backup controllers (a methodology for composing such lists is developed; as a result, the converging state of the network can improve significantly, depending on the selected heuristic).

As stated previously, this paper does not aim to develop a new optimized placement algorithm based on a given particular metric, but to consider the multi-criteria aspect of the problem and attempt to find an overall optimization.

III. METRICS AND ALGORITHMS- SUMMARY

This section summarizes some typical metrics and objectives of the optimization algorithms for controller placement. The overall goal is to optimize the Control Plane performance. Note that, given the problem complexity, the set of metrics and algorithms discussed below is not representing an exhaustive view. Considering a particular metric (criterion) an optimization algorithm can be applied, [10][11][14][19]. The goal of this paper is not to discuss details of such particular algorithms but searching a global optimization method. We only outline here their objectives and emphasize limitations of some particular cases.

A. *Performance-only related metrics (failure-free scenarios)*

The network is represented by an undirected graph G(V, E) where V is the set of nodes, n=|V| is the number of nodes and E is the set of edges. The edges weights represent an additive metric (e.g., *propagation latency* [11]). It is assumed that controller's locations are the same as some of the network forwarding nodes.

A simple metric is d(v, c): *shortest path* distance from a forwarder node $v \in V$ to a controller $c \in V$. In [11], two kinds of latencies are defined, for a particular placement C_i of controllers, where $C_i \subseteq V$ and $|C_i| \leqslant |V|$. The number of controllers is limited to $|C_i| = k$ for any particular placement C_i. The set of all possible placements is denoted by $C = \{C_1, C_2, \ldots\}$. One can define, for a given placement C_i:

Average_latency:

$$L_{avg}(C_i) = \frac{1}{n} \sum_{v \in V} \min_{c \in Ci} d(v,c) \quad (1)$$

Worst_case_latency :

$$L_{wc} = \max_{v \in V} \min_{c \in C_i} d(v,c) \quad (2)$$

The optimization algorithm should find a particular placement C_{opt}, where *either average latency or the worst case latency is minimal.*

Figure 1 shows a simple example of a network having six nodes.

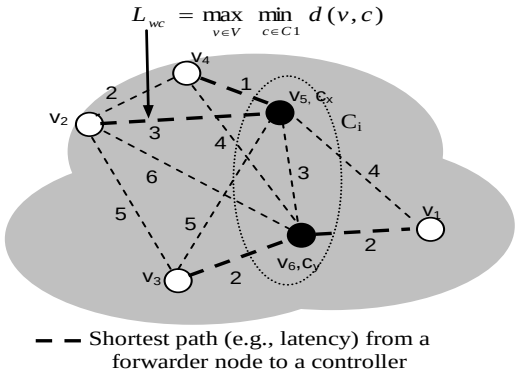


Figure 1. Simple network example of controller placement and nodes-to -controller assignment: v= forwarder node; c = controller; C_1= { [c_x_in_v_5 (v_5, v_2, v_4)], [c_y_in_v_6(v_6, v_1, v_3)]}

Note that Figure 1 could represent not quite the real network infrastructure, but an *overlay*, in which only a

subset of the total number of real nodes appear. This assumption makes the model more general in the sense that a hybrid network could be supposed as being the infrastructure supporting the overlay. A subset of nodes (routers) can be SDN forwarders and in some of them SDN controllers could be hosted. Two controllers $\{c_x, c_y\}$ can be placed in any location of the six nodes, e.g. in $\{v_5, v_6\}$. This placement instance is denoted by C_1. The distances between different nodes (overlay paths) are marked on the graph.

How to assign the forwarders to controllers? If one considers C_1 and the "shortest path" as a criterion of selection, then the allocation of forwarders to controller will be:

C_1: $c_x = v_5$ (v_5, v_2, v_4), $c_y = v_6$(v_6, v_1, v_3)

Note that a particular network node could play the both roles , as forwarder and controller.

Some limitations of this optimization process are:

- No reliability awareness: the metrics are pure distances, which in the simplest case are considered as being static values - despite that *delay* is usually a dynamic value in IP networks.
- There is no upper limit on the number of v nodes assigned to a controller; too many forwarders controlled by the same controller could exist, especially in large networks.

Other metric possible to be considered in failure-free case is *Maximum cover* [11][20]. The algorithm should find a controller placement, as to *maximize the number of nodes within a latency bound*; i.e., to find a placement of k controllers such that they cover a maximum number of forwarder nodes, while each forwarder must have a limited latency bound to its controller.

All metrics and algorithms described above do not take into account the inter-controller connectivity, so their associated optimizations can be seen as being partial.

B. Reliability aware metrics

Several studies consider more realistic scenarios in which controller failures or network links/nodes failures might exist. The optimization process aims now to find trade-offs (related to failure-free scenarios) in order to preserve a convenient behavior of the overall system in failure cases.

(1) Controller failures (cf)

The work [14] observes that the node-to-controller mapping can change in case of controller outages. So, a realistic latency-based metric should consider both the distance to the (primary) controller and the distance to other (backup) controllers. For a placement of a total number of k controllers, in [14] the failures are modeled by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 of up to k - 1. The resulting maximum latency will be:

Worst_case_latency_cf :

$$L_{wc-cf} = \max_{v \in V} \max_{C_i \in C} \min_{c \in C_i} d(v, c) \quad (3)$$

The optimization algorithm should find a placement which *minimizes the expression* (3).

Commenting the placement results based on the metric (1) or (2), one can observe that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders nodes. However, when attempting to minimize the expression (3) (and considering worst case failure), the controllers tend to be placed in the centre of the network. Thus, even if all controllers (except for one) fail, the latencies are still satisfactory (numeric examples are given in [14]).On the other side, one can criticize such an approach if applied to large networks; the scenario supposed by the expression (3) is very pessimistic; it is more realistic that a large network will be split in some regions/areas, each served by a primary controller; then some lists of possible backup controllers can be constructed for each area, as proposed in [19].

The conclusion is that a trade-off exists, between the placements optimized for the failure free case and those including controller failures. It is a matter of operator policies to assign weights to different criteria, before deciding (based on multiple criteria) the final selection of placement solution.

(2) Nodes/links failures (Nlf)

Links or nodes failures might produce network disruptions; some forwarders could have no more access to any controller. Therefore, an optimization objective could be to find a controller placement, which minimizes the number of nodes possible to enter into controller-less situations, in various scenarios of link/node failures. A realistic assumption is to limit the number of simultaneous failures at only a few (e.g., two [14]). If more than two arbitrary link/node failures happen simultaneously, then the topology can be totally disconnected and optimization of controller placement would be no more efficient.

For *any given placement* C_i of the controllers, an additive integer value metric $Nlf(C_i)$ could be defined, as below:

- consider a failure scenario denoted by f_k, with $f_k \in F$, where F is the set of all network failure scenarios (suppose that in any instance scenario, at most two link/nodes are down);
- initialize $Nlf_k(C_i) = 0$; then for each node $v \in V$, add one to $Nlf_k(C_i)$ if the node v has no path to any controller $c \in C_i$ and add zero otherwise;
- compute the maximum value (i.e., consider the worst failure scenario). We get:

$$Nlf(C_i) = \max Nlf_k(C_i) \quad (4)$$

where *k* covers all scenarios of F.

The *optimization algorithm* should find that *placement which minimizes (4).* It is naturally expected that

increasing the number of controllers, will decrease the Nlf value. *We also observe that the optimum solution based on the metric (4) could be very different from those provided by the algorithms using the metrics (1) or (2).*

(3) Load balancing for controllers

A well designed system would require roughly equal load on all controllers, i.e., a good balance of the node-to-controller distribution. A metric can be defined to measure the degree of imbalance *Ib(C_i)* of a given placement C_i as the *difference between the maximum and minimum number of forwarders nodes assigned to a controller.* If the failure scenarios set S is considered, then the worst case should evaluate the maximum imbalance as:

$$Ib(C_i) = \max_{s \in S} \{ \max_{c \in C_i} n_c^s - \min_{c \in C_i} n_c^s \} \quad (5)$$

where n_c^s is the number of forwarder nodes assigned to a controller c. Equation (5) takes into account that in case of failures, the forwarders can be reassigned to other controllers than the primary ones and, therefore, the load of those controllers will increase. An *optimization algorithm* should find that *placement which minimizes the expression (5).*

(4) Multiple-path connectivity metrics

One can exploit the possible multiple paths between a forwarder node and a controller [19], hoping to reduce the frequency of controller-less events, in cases of failures of nodes/links. The goal is to maximize connectivity between forwarding nodes and controller instances. The metric is defined as:

$$M(C_i) = \frac{1}{|V|} \sum_{c \in C_i} \sum_{v \in V} ndp(v,c) \quad (6)$$

In (6), *ndp(v,c)* is the *number of disjoint paths* between a node v and a controller c, for an instance placement C_i. An *optimization algorithm should find the placement C_{opt} which maximizes M(C_i).*

C. Inter-controller latency (Icl)

The inter-controller latency has impact on the response time of the inter-controller mutual updating. For a given placement C_i, the *Icl* can be given by the maximum latency between two controllers:

$$Icl(C_i) = \max d(c_k, c_n) \quad (7)$$

Minimizing (7) will lead to a placement with controllers close to each other. However, this can increase the forwarder-to-controller distance (latency) given by (1) or (2). Therefore, a trade-off is necessary, *thus justifying the necessity to apply some multi-criteria optimization algorithms, e.g., like Pareto frontier - based ones.*

D. Constraints

Apart from defining the metrics, the controller placement problem can be subject to different constraints. For instance, in [19], the input data for the optimal controller placement algorithm consists in the graph G(V,E) information, set of possible controller instances C, request demand of a network device, each controller capacity, and a backup capacity for each controller. *Integer Linear Programming (ILP)* – based algorithm is applied; here the constraints can be split into three classes: placement-related, capacity related and connectivity-related. In general other limits can be defined, e.g., on maximum admissible latency, ratio number controller/trivial nodes, pre-defined regions for controllers, etc. They should be included in the respective algorithms.

IV. Multi-citeria optimization Algorithm

Sections II and III have shown that several criteria of optimum can be envisaged when selecting the best controller placement in a WAN. While particular metrics and optimization algorithms can be applied (see Section III), we note that some criteria lead to partially contradictory placement solutions. What approach can be adopted? The answer can be given by adopting a multi-objective optimization based on *Multi-Criteria Decision Algorithms (MCDA).* The good property of MCDA is that it allows selection of a trade-off solution, based on several criteria. Note that, partially, such an approach has been already applied in [14], for some combinations of the metrics defined there (e.g., max. latency and controller load imbalance for failure-free and respectively failure cases).

A. Reference level MCDA

This paper proposes to apply MCDA, as a general way to optimize the controller placement, while considering not only a single metric but an arbitrary number of them.

The multi-objective optimization problem [12][13] is to minimize $\{f_1(x), f_2(x), \ldots, f_m(x)\}$, where $x \in S$ (set of feasible solutions), $S \subset R^n$. The *decision vector* is $x = (x_1, x_2, \ldots, x_n)^T$. There are ($m \geqslant 2$) possibly conflicting objective functions $f_i : R^n \rightarrow R$, i = 1, ..m, and *we would want to minimize them simultaneously (if possible).* In controller placement problem we might have indeed some partially conflicting objectives (e.g., to minimize the inter-controller latency and the forwarder-controller latency).

One can define *Objective vectors* as images of decision vectors. The objective (function) values are given by $z = f(x) = (f_1(x), f_2(x), ..., f_m(x))^T$. We denote as feasible objective region W = f (S) = image of S in the objective space.

Objective vectors are optimal if none of their components can be improved without deterioration to at least one of the other components.

A decision vector x_ ∈ S is named Pareto optimal [12] if it does not exist another x ∈ S such that $f_i(x) \leqslant f_i(x\_)$ for all i = 1, . . . , m and $f_i(x) < f_j(x\_)$ for at least one index j.

We adopt here the MCDA variant called *reference level decision algorithm* [13]. It has the advantage to allow selection of the optimal solution while considering normalized values of different criteria (metrics).

We use a simplified notation:

- identify the solutions directly by their images in the objectives space R^m,
- decision parameters/variables are: v_i, i = 1, ..m, with ∀i, $v_i \geqslant 0$,
- image of a candidate solution is $Sl_s=(v_{s1},v_{s2},...,v_{sm})$, represented as a point in R^m ,

S = number of candidate solutions.

Note that the value ranges of decision variables may be bounded by given constrains. The optimization process consists in selecting a solution satisfying a given objective functions and conforming a particular metric.

The basic *reference level algorithm* [13], defines two reference parameters:

- r_i=*reservation level*=the upper limit for a decision variable, which the decision variable v_i of a solution should not cross;
- a_i=*aspiration level*=the lower bound beyond which the decision variable (and therefore the associate solutions) are seen as similar.

Without loss of generality one may apply the definitions of [13], where for each decision variable v_i there are defined two values named r_i *and* a_i, by computing among all solutions s = 1, 2, ..S:

$$r_i = \max [v_{is}], s = 1, 2, ..S$$
$$a_i = \min [v_{is}], s = 1, 2, ..S \quad (8)$$

In [13], modifications of the decision variables are proposed: *replace each variable* with *distance from it to the reservation level*: $v_i \rightarrow r_i - v_i$; (increasing v_i will decrease the distance); normalization is also introduced, in order to get non-dimensional values, which can be numerically compared. For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s, i \quad (9)$$

The factor $1/(r_i-a_i)$ - plays also the role of a weight. The variable having high dispersion of values (max – min) will have lower weights, and so, greater chances to determine the minimum in the next relation (10). In other words, less preference is given to those decision variables having close values to each other (among different candidate solutions), i.e., if the values *min, max* are close enough, then it does not matter which solution is chosen by considering the respective decision variable.

The basic algorithm steps are:

Step 0. Compute the matrix $M\{v_{si}'\}$, s=1…S, i=1…m

Step 1. Compute for each candidate solution *s*, the minimum among all its normalized variables v_{si}':

$$\min_s = \min \{v_{si}'\}; i = 1...m \quad (10)$$

Step 2. Make selection among solutions by computing:

$$v_{opt} = \max \{ \min_s \}, s = 1, ..S \quad (11)$$

Note that the formula (10) selects for each candidate solution *s*, the worst case, i.e., that solution being closest to the reservation level (after searching among all decision variables). Then the formula (11) selects among the solutions, the best one, i.e., that having the highest value of the normalized parameter.

This v_{opt} is the optimum solution, i.e., the MCDA selects the best value among those produced by the Step 1. In the case that several equal values exist in the Step 2, a random selection can be adopted or some other additional discrimination criterion. Note also, that it is no problem for the Step 2, to consider more than one solution, i.e., a set of several quasi-optimum solutions can be selected.

B. MCDA- Controller placement optimization

In this section, we *apply the reference level algorithm to the controller placement problem*. However, we modify the basic algorithm to be better adapted to controller placement problem, due to following remarks:

(1) The step 2 *compares values coming from different types of parameters/metrics* (e.g., max. latency, load imbalance, etc.) having different nature and being independent or dependent on each other. The normalization still allows them to be compared in the *max{ }* formula. *This is an inherent property of the basic algorithm.*

(2) However, the network provider might want to apply different policies when deciding the controller placement. In such policies, some decision variables (or metrics) could be "more important" than others. For instance, in some cases, the performance is more important, in others high resilience is the major objective.

A simple modification of the algorithm can support a variety of provider policies. We propose a modified formula:

$$v_{si}' = w_i (r_i - v_{si})/(r_i - a_i) \quad (12)$$

where the factor $w_i \in (0,1]$ represents a weight (priority) that can be established from network provider policy considerations, and can significantly influence the final

selection. Note that a lower value of w_i actually represents a higher priority of that parameter in the selection process.

The controller placement problem solving (given the graph, link costs/capacities, constraints, desired number of controllers, etc.) is composed of two macro-steps:

(1) *Macro-step1*: Identify the parameters of interest, and compute the *values of the metrics for all possible controller placements*, using specialized algorithms and metrics (1) - (7). In other word this step will produce the set of candidate solutions (i.e., placement instances).

This procedure could be time consuming (depending on network size) and, therefore, performed off-line [11].

(2) *Macro-step2*: MCDA

- define reservation and aspiration levels for each decision variable;
- eliminate those candidates having out of range parameter values defined by the reservation level;
- define appropriate weights (see formula (12)) for different decision variables - depending on the high level policies applied by the operator;
- compute the normalized variables (formula (12))
- run the Step 0, 1 and 2 of the MCDA algorithm (formulas (10) and (11)).

The decision variables can be among those of Section III, i.e.:

Average (1) or *worst* (2) *case latency* (failure-free case);

Worst_case_latency_cf (3);

Nodes/links failures (Nlf) (4);

Controller Load imbalance (5);

Multi-path connectivity metric (6);

Inter-controller latency (7).

For a particular problem, a selection of relevant variables should be done. For instance, in a high reliable network environment one could consider only failure free metrics.

C. Numerical example – MCDA optimization

A simple but relevant example is exposed to illustrate the MCDA power, based on the network in Figure 1. Suppose that for this network the metrics of interest and decision variables are (see Section III) on:

d1: *Average latency* (1), (failure-free case);

d2: *worst latency (2,)* (failure-free case);

d3: *Inter-controller latency (7).*

The reference levels are defined as in formula (8) and we propose: r_1=3, a_1=0; r_2=6, a_2=0; r_3=6, a_3=0.

Several placement samples can be considered:

C_1= {[c_x_in_v_5 (v_5, v_2, v_4)], [c_y_in_v_6(v_6, v_1, v_3)]}

C_2= {[c_x_in_v_5 (v_5, v_1, v_2, v_4)], [c_y_in_v_3(v_3, v_6)]}

C_3= {[c_x_in_v_3 (v_3,v_2)], [c_y_in_v_6(v_6, v_1, v_4, v_5)]}

C_4= { [c_x_in_v_4 (v_4,v_2, v_5)], [c_y_in_v_6(v_6, v_1, v_3)]}

1. MCDA with equal priorities for d1=1, d2=1, d3=1, The values of the metrics are computed using equations (1), (2) and respectively (7) for each placement: C_1, ..C4.

A matrix M(3x4) is computed using the formulas (9). MCDA is applied by using formulas (10), (11). The final result is: *C_1 = the best placement.* Looking at Figure 1, we indeed can see that this placement is a good trade-off between node-controller latency and inter-controller latency.

2. MCDA with different priorities for i.e. d1=1, d2=0.5, d3=1, i.e., the worst case latency d2 has highest priority, i.e., the solution minimizing the worst case controller - forwarder latency with high priority is desired. After re-computing the matrix M and applying MCDA equations (1), (11), we find *C_4 = the best placement*. Indeed, we see in Figure 1 that worst case latency (node-controller) is minimized, however, the inter-controller latency is higher than in C_1.

These examples proved how different provider policies can bias the algorithm results.

V. USE CASE STUDIES

A proof of concept simulation software program has been constructed to validate the above MCDA – based controller assignment procedure. This preliminary version of the program has been written in Python language [21] and uses the NetworkX software package [22] for the creation, manipulation and study of network graphs. The program has two running modes.

Static: in these modes the inputs are:

- the network (overlay) topology graph and link costs (it is supposed an additive metric representing the estimated delays on overlay network links);
- the number of controllers wanted;
- decision parameters – e.g., d1, d2, d3 of the previous section;
- priorities/weights assigned to the decision variables that comply with the network provider policy;
- the possible placement of the controllers (i.e., candidate solutions of the MCDA) (considering them as results of some other algorithms)

Dynamic: in this mode the following parameters can be selected:

- total number of network nodes (N)
- desired number of controllers
- the link costs (e.g., randomly assigned).

The program computes all possible placements and then selects the best solution based on weighted MCDA algorithm.

The program parses user arguments, constructs the weighted graph and associated candidate solutions,

computes Dijkstra's shortest path lengths between all nodes in the graph, and then applies the MCDA controller placement optimization algorithm. The flowchart of the program is depicted in Figure 2.

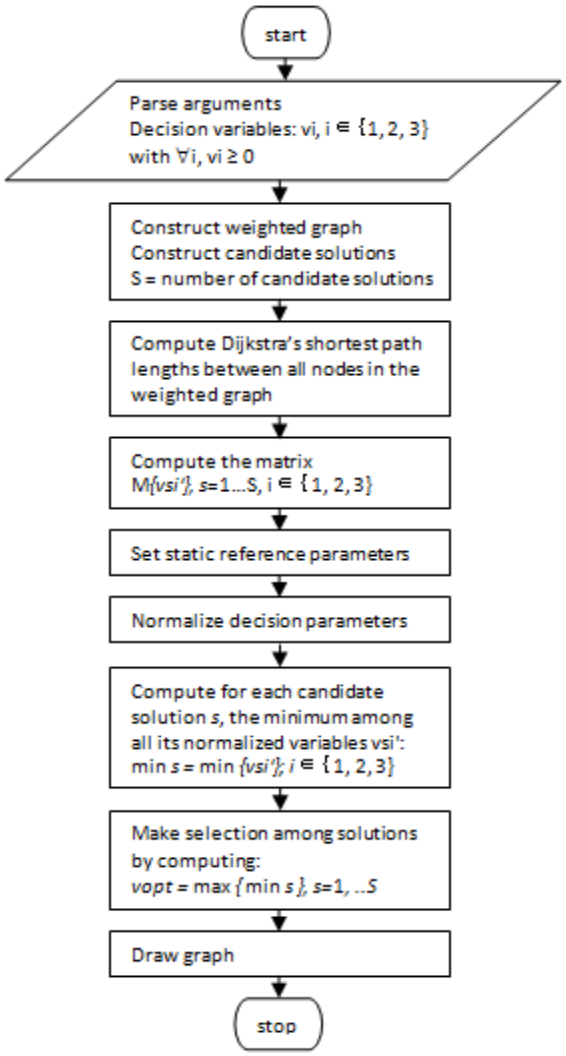


Figure 2. Flowchart for MCDA controller placement simulation program.

Generally the order of complexity of an algorithm is an important issue to be considered if its implementation is targeted. In our case the following particular characteristics of the optimization problem are valid :

(a) - this optimization computation has no real time requirements; it could be performed offline for a given fixed network (topology and costs are known or estimated).

(b) - the complexity order of the Dijkstra part of the algorithm is the well known to be $O(|V|^2)$, or $O(|E| + |V| \log|V|)$, if a more efficient implementation is chosen, where |V | is the number of nodes and |E| is the number of edges.

(c) the evaluation of the metrics (1) –(7) supposes that for every particular mono-criteria algorithm, all possible placements should be considered. For large networks this is given in the worst case by the number of combinations C_n^k, (n= number of nodes; k = number of controllers) which increases very much with the number n of network nodes (usually it is true that k<<n). However this problem is common to all algorithms and it is not particular to the algorithm developed in this paper.

(d) the MCDA algorithm itself, has to construct a matrix having the dimension NL*NC, where:

-NL is the number of lines - equal to the number of decision variables (e.g. three in the example given in the Section IV.C);

- NC is the number of columns – equal to the number of candidates' solutions (variants of controller placements).

The number NL is usually small (e.g., NL=7 in this study). The number NC could be large, given by C_n^k, which is the number of possible combinations of controller placements. The Stirling formula $n! \approx (2\pi n)^{1/2} (n/e)^n$ shows a strong increase of the number of controller placements C_n^k with n. However, some practical considerations and/or policies could reduce significantly the actual number of combinations to be considered. It is expected that large networks will be split in some disjoint regions having significantly less than n nodes and a number of reduced number k_R controllers will be allocated for a given region. In the work [11] the authors even say that "…in many medium-size networks, the latency from every node to a single controller can meet the response-time goals of existing technologies, such as SONET ring protection…". If the number of regions is R, with R > 1then the number of nodes n will be reduced at n/R, strongly decreasing the number of combinations. Some other restrictions imposed from policy considerations could also reduce the total number of nodes to be considered in the formulas (1) – (7).

The interface for running the simulation program is presented below.

```
[b38632@localhost mcda]$ python mcda.py --help
usage: mcda.py [-h] [-a A] [-w W] [-i I] [--dynamic] [-n N] [-c C]

Multi-criteria optimization algorithm

optional arguments:
  -h, --help  show this help message and exit
  -a A        Average latency - failure free scenario. Expects a weight
              (priority) in interval (0, 1].
  -w W        Worst case latency - failure free scenario. Expects a weight
              (priority) in interval (0, 1].
  -i I        Inter controller latency. Expects a weight (priority) in
              interval (0, 1].
  --dynamic   Generate dynamic undirected graph
  -n N        Number of graph nodes. Valid only in dynamic mode.
  -c C        Number of controllers in graph. Valid only in dynamic mode.
              Allowed values are between N/3 and N/7
[b38632@localhost mcda]$
```

Some samples of simulation results are given below.

Figure 3 shows the results obtained for the network presented in Figure 1, while decision variables are d1, d2, d3 defined in Section IV. In this use case, the highest priority is given to the worst case latency parameter (d2 = 0.5), while the two others have d1=d3=1. One can see that the best solution selection is the setup C4, i.e., the same results as analytically estimated in Section IV.

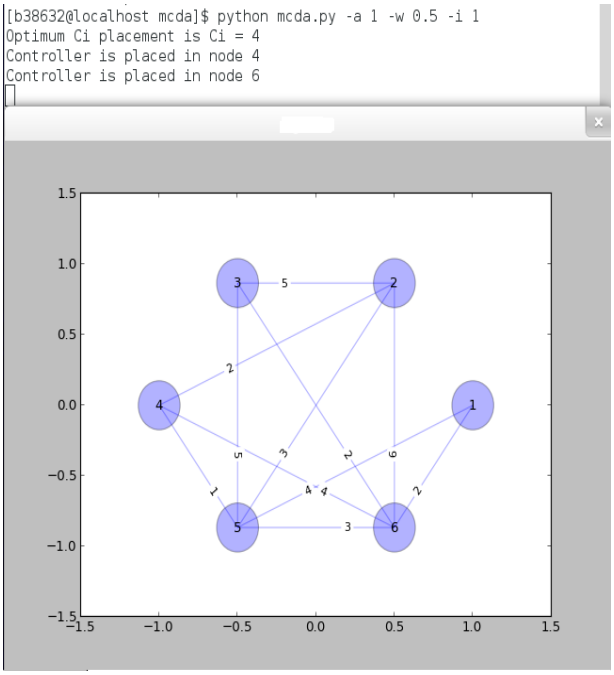


Figure 3. Simulation results for Figure 1 network; *d1=1, d2=0.5, d3=1.*

Figure 4 shows another instance of use case, where the graph is dynamically generated, with N= 7 nodes and k=2 controllers. The decision parameters have equal priorities: d1 = 1, d2 = 1, d3 = 1. The best placement is denoted by C_3 and places the controllers in the nodes having the number 0 and 4 respectively. Note that the network of Figure 4 is a full mesh one. This is the effect of considering the overlay of paths. However, the algorithm and the program can work as well with partial mesh overlay graph.

Figure 5 shows a quantitative extension for a larger graph – dynamically generated with N=14 nodes and k = 5 controllers. The picture exposes a symmetrical figure due to full mesh connectivity between nodes. The link costs are randomly generated. The total number of possible placements is C_N^k. The MCDA selects the placement no. 55, indicated in the text associated to the figure.

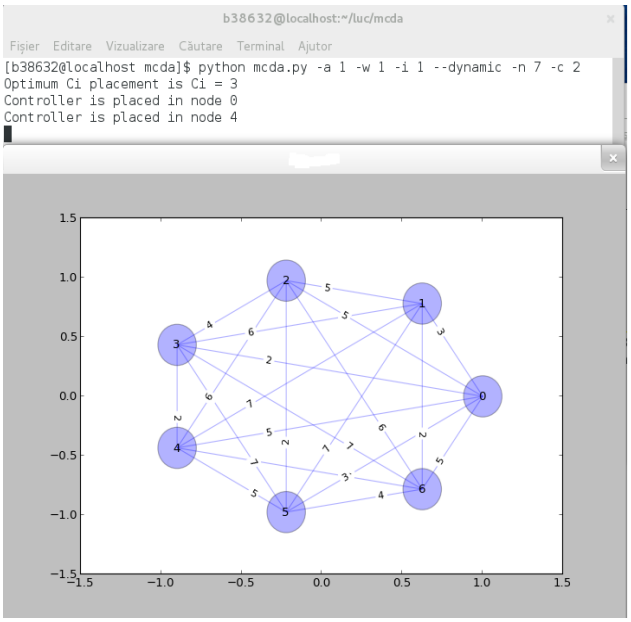


Figure 4. Simulation results for a network having N= 7, network; *d1=1, d2=1, d3=1.*

VI. Conclusions

This paper presented a study on using multi-criteria decision algorithms (MCDA) for final selection among several controller placements solutions in WAN SDN, while considering several weighted criteria. The MCDA quality is that it can produce a tradeoff (optimum) result, while considering several criteria, part of them even being partially contradictory.

A simulation program has been created to demonstrate the validity of results. The topology and link costs are generally overlay ones, and can be introduced in a particular way or randomly generated.

The method proposed is generic enough to be applied in various scenarios (including failure-free assumption ones or reliability aware), given that it achieves an overall optimization, based on multiple metrics supported by the reference model MCDA. Different network/service provider biases can be introduced in the selection process, by assigning policy-related weights to the decision variables. This simple algorithm modification creates a rather powerful tool to bias the selected solution, as to respond to the provider policy.

Future work will be done to apply the method proposed to very large networks - real life case studies (e.g., from Internet Topology zoo, [16]) and comparing the quality of trade-offs when defining different weights to decision variables.

```
[b38632@localhost mcda]$ python mcda.py -a 1 -w 1 -i 1 --dynamic -n 14 -c 5
Optimum Ci placement is Ci = 55
Controller is placed in node 0
Controller is placed in node 1
Controller is placed in node 3
Controller is placed in node 4
Controller is placed in node 5
```

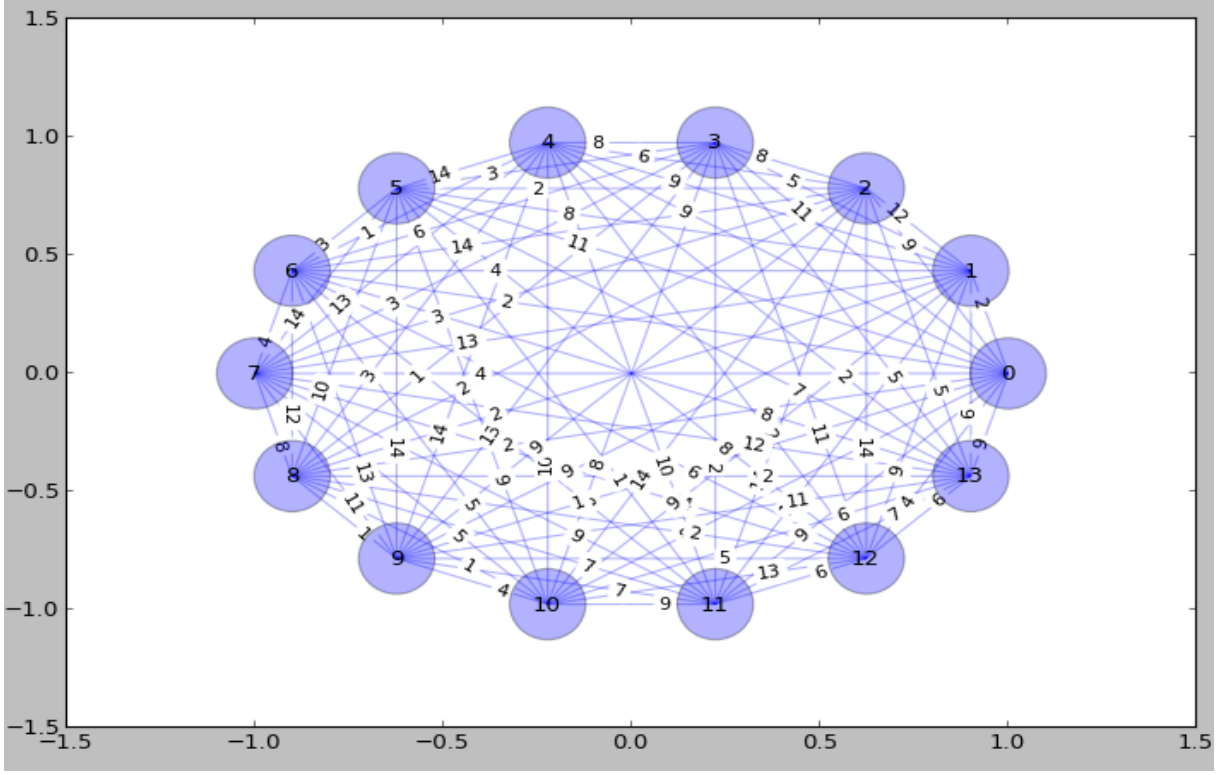


Figure 5. MCDA for a dynamic graph with 14 nodes (5 controllers) and equal priorities; *d1=1*, *d2=1*, *d3=1*.

REFERENCES

[1] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On Multi-controller Placement Optimization in Software Defined Networking - based WANs," The International Symposium on Advances in Software Defined Networks SOFTNETWORKING 2015, 2015 - Barcelona, Spain, http://www.iaria.org/conferences2015/SOFTNETWORKING.html

[2] B. N. Astuto, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," Communications Surveys and Tutorials, IEEE Communications Society, (IEEE), 2014, 16 (3), pp. 1617 – 1634.

[3] "Software-Defined Networking: The New Norm for Networks" ONF White Paper 04.2012; retrieved: 02.2015 https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[4] "SDN: The Service Provider Perspective," Ericsson Review, 21.02.2013. retrieved: 02.2015 http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2013/er-software-defined-networking.pdf.

[5] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, "On Scalability of Software-Defined Networking," IEEE Comm. Magazine, February 2013, pp. 16-141,.

[6] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in European Workshop on Software Defined Networks (EWSDN), Darmstadt, Germany, October 2012.

[7] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010.

[8] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010.

[9] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," Proc. HotSDN '12 Wksp., 2012.

[10] H. Yan-nan, W. Wen-dong, G. Xiang-yang, Q. Xi-rong, C. Shi-duan, "On the placement of controllers in software-defined networks," ELSEVIER, Science Direct, vol. 19, Suppl.2, October 2012, pp. 92–97, http://www.sciencedirect.com/science/article/pii/S100588851160438X.

[11] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, 2012, pp. 7–12.

[12] J. Figueira, S. Greco, and M. Ehrgott, "Multiple CriteDecision Analysis: state of the art surveys," Kluwer Academic Publishers, 2005.

[13] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization". Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer-Verlag, pp. 468–486.

[14] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in ITC, Shanghai, China, 2013.

[15] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/.

[16] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011.

[17] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in GLOBECOM 2011, 2011.

[18] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability aware controller placement for software-defined networks," in Proc. IM. IEEE, 2013, pp. 672–675.

[19] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, M. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability," IEEE Global Comm. Conference (GLOBECOM); 12/2014.

[20] D. Hochba "Approximation algorithms for np-hard problems", ACM SIGACT News, 28(2), 1997, pp. 40–52.

[21] https://www.python.org/doc/essays/blurb/

[22] https://networkx.github.io/

Network Partitioning Problem for Effective Management of Multi-domain SDN Networks

Hidenobu Aoki, Norihiko Shinomiya
Graduate School of Engineering
Soka University
Tokyo, Japan
Emails: aoki39h@gmail.com, shinomi@soka.ac.jp

***Abstract*—In Software-Defined Networking, a network with distributed controllers can be partitioned into sub-networks as controller's administrative domains. Network partitioning could affect various aspects of network performances, such as controller load and reliability of controller's domains because network resources and topology are logically divided into sub-networks. By focusing on network partitioning, this paper handles the issue as a mathematical problem based on graph clustering and analyzes effective network partitioning methods with different indicators related Software-Defined Networking. The simulation results indicate the effectiveness of our clustering method in terms of the load balance of controllers and the reliability of controller domains.**

***Keywords*—Software-Defined Networking; distributed controllers; network partitioning; graph clustering.**

I. Introduction

This paper extends the conference paper presented in SOFTNETWORKING 2015 [1], which evaluates clustering algorithms from diversified standpoints of the network partitioning in Software-Defined Networking (SDN).

SDN has been emerging as a new networking paradigm. The fundamental concept of SDN is to achieve programmable networking by separating the control and the data planes in an individual network device, such as a switch and router [2]. In an SDN network, a controller is in charge of generating data forwarding rules. In contrast, network devices in the data plane are responsible for forwarding data according to the rules. This centralized architecture where a controller manages network devices enables network operators to dynamically configure network devices and to flexibly manage their networks. Currently, its applications have been extended to various types of networks, such as campus, datacenter, and carrier networks [3].

However, it has been discussed that a single controller has raised scalability and reliability issues. In large-scale networks, the load could converge on a single controller even though its processing capacity could be limited [4]. Moreover, if a failure occurs on a single controller, an entire network managed by the controller could take a risk of breakdown [5]. Furthermore, in wide-area networks, it could cause communication latency because some switches may be located far away from the controller. As a result, it might not be feasible to process events requiring real-time operations, such as failure recovery [6]. To handle those issues on a single controller, it has been studied to deploy multiple controllers over a network as one of main SDN-related research topics [7][8].

In such an SDN network with distributed controllers, there are mainly two types of control: the hierarchical and flat controls [9]. The hierarchical control is to organize controller's functions vertically. For example, a function dealing with flow setup messages or maintaining the states of network devices on data plane while another exchanges network information with other controllers to keep a global view of a network.

In contrary, the flat control is to partition a network into sub-networks, and controllers are assigned to one of them as their administrative domains. Because of the network partitioning, each sub-network can be managed independently by a controller. Hence, it could reduce the overall complexity of the whole network management and the computational load of each controller as well as handling flow setup requests faster and more efficiently. Moreover, it would be preferable to limit the scope of network operations for the deployment of new technology or infrastructure, such as adding or relocating network devices. Furthermore, when a controller failure occurs, it could alleviate to spread its negative effects to the rest of the network [10].

On partitioning a network, various aspects of the network performance could be affected since network resources and topology are logically divided into sub-networks. For instance, the load of controllers would be regarded as one of the major issues. Generally, the controller load is originated from network provisioning and status collection overhead within a domain and collaboration load among controllers, and these overheads could depend on how network resources are distributed in sub-networks [11]. Additionally, it would be necessary to consider the reliability of sub-networks as well as the controller load. In reality, controllers will be located in the same position as switches, and the control and data messages are flowed through the same communication links in the in-band control model. As a result, it would be desirable to partition a network so that switches in each sub-network have multiple paths reaching to controllers and other switches in case of link failures.

Therefore, this paper focuses on the network partitioning which is related to the flat control. In order to analyze its effective way, we provide four clustering algorithms and evaluate them based on the different indicators associated with

SDN networks.

The organization of this paper is as follows: Section II introduces the related work of the hierarchical control of distributed SDN controllers and network partitioning. Section III explains the layered architecture of control plane and addresses the issues of network partitioning. Section IV provides the definitions of the graphs and formulates Network Partitioning Problem (NPP). Section V details clustering algorithms as solutions of network partitioning. Section VI describes the simulation results and discussions, and Section VII concludes this paper.

II. Related Work

In this section, the related work of the hierarchical control of distributed SDN controllers and network partitioning are presented.

A. Hierarchical Control Plane

Onix [12] describes network topology as a graph. Each partitioned network is contracted to a logical node and used as a unit to share network information among controllers. This enables a controller to communicate with other controllers without knowing specific network states and topology of other partitioned networks. In this way, the reduction of the amount of network information possessed by a controller can be achieved.

Kandoo [13] provides a hierarchical control method consisting of the root controller and some local controllers. The root controller manages all local controllers and is responsible for the events which requires information over the whole network. On the other hand, local controllers deal with the local events, such as flow setups and network statistics collections of a local network. This layered control defines the scope of operations to process different requests efficiently, which could offload the burden of the root controller.

Although the ideas of hierarchical control plane have been proposed in those researches, they do not discuss how to partition a network to decide controller's domains.

B. Network Partitioning

On partitioning a network, the major issue to consider would be the controller load. The controller load is generally thought to be the overhead to configure and mange network devices within domains. Simply, the more switches a controller needs to configure, the heavier load it is imposed on the controller. Thus, it has been studied to partition a network aiming at balancing the controller load by equalizing the number of switches in domains [14][15].

Contrary, Yao et al. in [16] argues that the importance of switches should be considered for load balance of controllers because it should differ from each other depending on networks. In the research, weight of the switch is given as the degree of nodes representing the number of flow setup requests required by the switch. Although the research considers the new metric for load balance of controllers, the focus of the research is where to place controllers to satisfy the metric and does not discuss how to partition a network in detail.

On the other hand, connectivity of a network is considered as one of the objectives for the controller placement problem [17][18]. Those researches aim to maximize the number of disjoint paths between a controller and switches or between switches so as to ensure the connection between them in case of link failure. Nevertheless, their approaches do not focus on network partitioning methods but controller locations to enhance reliability of domains.

Our previous work [1] proposes a network partitioning method whose objective is to minimize the number of inter-domain links in terms of the reduction of the controller load in sharing topology information. However, the work does not evaluate the load balance of controllers and reliability of controller domains. Thus, as the extension of our previous work, this paper provides clustering algorithms and evaluates the load balance of controllers, the reliability of each domain as well as inter-control load.

III. Layered control plane

This section discusses the architecture of the layered control plane and addresses the issues on the network partitioning.

A. Definitions of Two-tier Control Plane

In an SDN network with multiple controllers, the network can be logically partitioned into sub-networks as controller domains. In each domain, a controller is mainly in charge of two roles: (1) control of switches in own domain and (2) federation of a whole network by communicating with other controllers. As a result, control plane can be layered in two tiers: the local and the federation tiers are responsible for (1) and (2), respectively.

B. Network Topology in Local and Federation Tiers

In the local tier, the local control function abstracts and possesses the network topology of each administrative domain as a local graph. Moreover, the local graph is contracted to a single node, which is used as a unit of communication with other controllers. In the federation tier, the global control function gathers the contracted nodes from all controllers and aggregates them to form a federation graph, which describes global network topology. Note that edges in a federation graph correspond to the edges between local graphs, which means inter-domain links are recognized as global topology information. Due to this topology contraction, it can be expected to reduce the amount of global network information shared among controllers.

Figure 1 illustrates an example of the layered control plane. In Figure 1, there are two domains described as local graphs 1 and 2. On the other hand, in the federation tier, the federation graph has two contracted nodes, and three edges correspond to the edges between the local graphs.

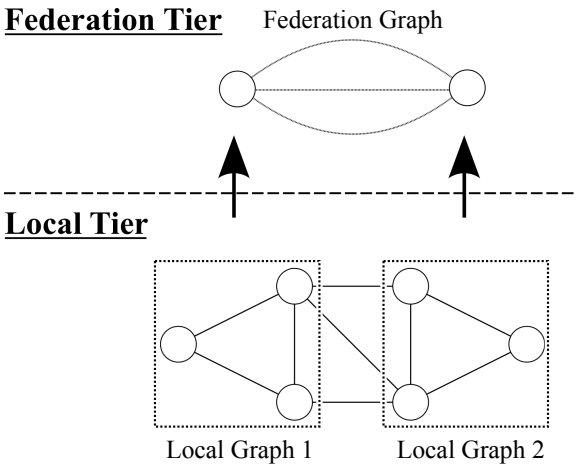


Fig. 1. Topology contraction in layered control plane.

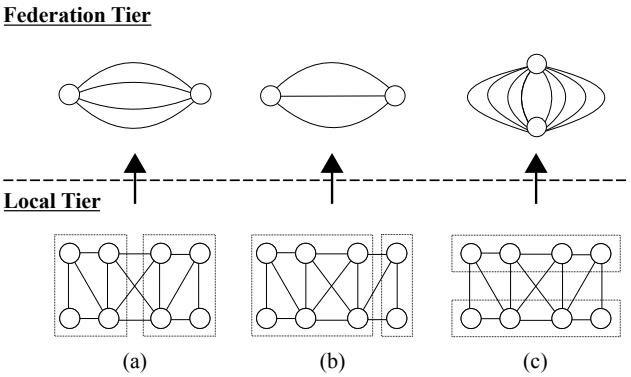


Fig. 2. Examples of network partitioning.

C. Issues on Network Partitioning

Because a network partitioning decides the distribution of network resources and topology into sub-networks, it would have an influence on the aspects of network operations and performances in SDN [14].

Figure 2 illustrates examples of the network partitioning in different ways. In Figure 2, comparing (a) and (b), there are four nodes in each domain in (a) although the domains in (b) contain different number of nodes: 6 and 2 nodes, respectively. This implies that the balance of the controller load to manage switches is determined by network partitioning.

Moreover, there are four edges in the federation graph (a) while the federation graph (c) has eight edges even though the domains in both (a) and (b) contain the same number of nodes. This might indicate the increase of the amount of shared information and collaboration overhead among controllers.

Additionally, although there are multiple paths between any pairs of nodes in domains of (a), there is only a single path connecting any pairs of nodes within domains of (c). As a result, when a link failure occurs on a link in the domains (c), even the communication to a switch in the same domain has to be via a path crossing another domain. This requires the extra inter-controller communication which leads to additional load on controllers.

As those examples imply, network partitioning would be an important issue to address for SDN networks. Therefore, this paper defines the problem to determine controller domains as Network Partitioning Problem (NPP) and analyzes effective way of the network partitioning.

IV. Problem Formulation

This section describes the graph definitions related to the layered control plane and formulates NPP.

A. Definitions

For a network graph $G = (V, E)$, a set of vertices V denotes network devices, such as routers and switches, and a set of edges E represents links between those devices. Considering the network partitioning, sub-networks called local graph are denoted as

$$G_1^l = (V_1^l, E_1^l), G_2^l = (V_2^l, E_2^l), \ldots, G_k^l = (V_k^l, E_k^l). \quad (1)$$

Note that we assume that a node can exclusively belong to a local graph. In the federation tier, on the other hand, a federation graph is defined as

$$G^f = (V^f, E^f), \quad (2)$$

where V^f indicates a set of the contracted nodes of the local graphs, and E^f represents that of the edges between the local graphs.

B. Clustering

In graph theory, clustering is a fundamental problem in mathematics and the applied science, which classifies the data into groups or categories. Graph clustering is defined as a task of grouping nodes in a graph into subsets called clusters [19]. Based on graph clustering, suppose that a local graph G_i^l in (1) is a cluster, and a set of local graphs $\mathbf{G^1}$ is denoted as a clustering:

$$\mathbf{G^1} = \{G_1^l, G_2^l, ..., G_k^l\}. \quad (3)$$

Generally, the desirable clustering is defined that there are many edges within each cluster called intra-cluster edges and relatively few edges between clusters referred to inter-cluster edges. Considering the network topology treated in the layered control plane, intra-cluster edges correspond to the edges in E_i^l, and inter-cluster edges are equivalent to the edges in E^f. In addition, a clustering having the less number of inter-cluster edges and the more number of intra-cluster edges is regarded as a preferable one [20].

C. Problem Formulation

In this paper, the primal objective of the network partitioning is to balance the load of controllers. The load of controllers would be composed of several factors, such as the management of local domains and the communication with other controllers for a global control. Among them, to handle flow setup requests might be one of the major loads of controllers because the potential bottlenecks of a network

will be bandwidth, memory, and processor of controllers when they receive many flow setup requests at a time [21]. In general, the amount of flow setup requests to a controller could increase depending on the number of switches it has to manage. However, since the importance of switches in a domain would be distinguished from each other, not only the number of switches in a domain but also the importance of the switches may also need to be considered as stated in [16]. Hence, as a measure of the importance of switches, this paper takes account of the switch weight. In our model, it is abstracted as the degree of nodes because the number of flow setup requests required by a switch could be related to its connection with other devices.

Here, for a graph G, let the weight of node u be given as its degree, $deg(u)$. Then, the total node weights in a domain is denoted as

$$D(G_i^l) = \sum_{u \in G_i^l} deg(u). \tag{4}$$

In order to evaluate the balance of intra-control load among controllers, the standard deviation of node weights in domains is calculated as follows:

$$\mathbf{G}_\sigma^\mathbf{1} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(D(G_i^l) - \overline{D(\mathbf{G^1})})^2}. \tag{5}$$

Note that $\overline{D(\mathbf{G^1})}$ stands for the mean value of the sum of node weights in a cluster, which is calculated as $\overline{D(\mathbf{G^1})} = \frac{\sum_{u \in V} deg(u)}{k}$. Therefore, the objective function of NPP is to find a clustering $\mathbf{G^1}$ such that $\mathbf{G}_\sigma^\mathbf{1}$ is minimized.

V. Clustering Algorithms

In this section, four clustering algorithms are provided as solutions of NPP.

A. Conductance Clustering

As one of clustering indices, conductance has been defined, which compares the sum of the number of inter-cluster edges and that of all edges yielded by a clustering [22]. By denoting a set of all edges that have their origin in G_i^l and their destination in G_j^l as $E(G_i^l, G_j^l)$, conductance of G_i^l is denoted as

$$\Phi(G_i^l) = \frac{|E(G_i^l, \mathbf{G^1} \setminus G_i^l)|}{\min(D(G_i^l), (D(\mathbf{G}^l \setminus G_i^l))}, \tag{6}$$

where $D(G_i^l)$ is the sum of node degree in G_i^l as defined in (4), and $D(G_i^l, \mathbf{G^1} \setminus G_i^l)$ is that of other clusters. Note that a cluster with smaller conductance represents a better cluster.

In general, finding a clustering with minimum conductance is known as NP-hard [20]. Hence, as proposed in [1], we construct Conductance clustering that chooses nodes one by one based on the conductance value shown in Algorithm 1. The algorithm begins with a random node. Then, one of neighbor nodes of the node, which the cluster obtains the smallest conductance value, is chosen. As this process, it expands the cluster by recursively choosing a neighbor node of the nodes in the cluster. If the number of nodes in the cluster reaches to an upper bound of the number of nodes in a cluster $\frac{|V|}{k}$, it starts again to create a new cluster with a random node, which has not belonged to any clusters.

Algorithm 1 Conductance Clustering.

Input: a graph G=(V, E), the number of node limitation $\frac{|V|}{k}$
1: Clustering $\mathbf{G^1} \leftarrow \phi$
2: $V' \leftarrow$ a list of nodes in a graph G
3: **while** $V' \neq \phi$ **do**
4: $G_i^l \leftarrow \phi$
5: Choose a node v_r from V' at random
6: Add v_r to G_i^l and remove v_r from V'
7: **while** the number of nodes in $G_i^l < \frac{|V|}{k}$ **do**
8: Find a neighbor node v_n of nodes in G_i^l which minimizes $\Phi(G_i^l)$
9: Add v_n to G_i^l and remove v_n from V'
10: **end while**
11: Add G_i^l to $\mathbf{G^1}$
12: **end while**
Output: $\mathbf{G^1}$

B. Spectral Clustering

Spectral clustering has been applied for various fields of data analysis [23][24]. It is a method to cluster data by using eigenvectors of the Laplacian matrix of a graph. The Laplacian matrix describes the structure of a graph where diagonal components represent the node degree, and others describe the adjacency relationship between corresponding nodes.

Here, suppose that the adjacency matrix of a graph G be defined as A_G with its elements determined by

$$A_G(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

On the other hand, the degree matrix of G is described as D_G which is a diagonal matrix composed of

$$D_G(u, u) = deg(u). \tag{8}$$

By using A_G and D_G, the Laplacian matrix of G is denoted as

$$L_G = D_G - A_G. \tag{9}$$

As described in Algorithm 2, Spectral clustering computes L_G of G and its eigenvectors. Then, based on the eigenvectors, it obtains a clustering by k-means clustering algorithm.

1) k-means Clustering: In Step 5 of Algorithm 2, k-means clustering is used, which is one of the most commonly used clustering algorithms [25]. For the k initial cluster centers obtained in Step 4 of Algorithm 2, each node of a graph is assigned to one of the closest cluster centers so as to satisfy the following function:

$$\text{Minimize} \sum_{i=1}^{k} \sum_{u \in G_i^l} |u - c_i|^2. \tag{10}$$

Algorithm 2 Spectral Clustering.

Input: a graph G=(V, E), the number of domains to create k.

1: Compute the Laplacian matrix L_G
2: Obtain the first k eigenvectors $u1, ..., u_k$ of L_G
3: Let $U \in \mathbf{R}^{n \times k}$ be the matrix containing the vectors $u1, ..., u_k$ as columns.
4: For $i = 1, ..., n$
let $y_i \in R^k$ be the vector corresponding to the i-th row of U.
5: Cluster the points $y_i \in \mathbf{R}^k$
with k-means clustering algorithm into cluster, $G_1^l, ..., G_k^l$.

Output: $\mathbf{G^1}$

C. Betweenness Centrality Clustering

As one of metrics for graph analysis, the betweenness centrality has been defined [26]. It implies how an edge is associated with the shortest paths between every pair of nodes in a graph. Thus, edges with high betweenness centrality are regarded to be important in the graph.

Here, the edge betweenness centrality is given by

$$C_B(u, v) = \sum_{s \neq u \neq v \neq t} \frac{\sigma_{st}(u, v)}{\sigma_{st}}, \tag{11}$$

where σ_{st} is the total number of shortest paths from node s to node t, and $\sigma_{st}(u, v)$ is the number of those paths that pass through edge (u, v).

In general, the edges connecting different groups tend to have high edge betweenness centrality. Thus, by removing these edges, the underlying group structure of a graph would be revealed [27]. Based on the idea, Girvan and Newman have developed a clustering algorithm shown in Algorithm 3. The algorithm begins with an initial cluster C_i containing all nodes in G. Then, it computes $C_B(u, v)$ for all edges in C_i and removes the edge with the highest $C_B(u, v)$ recursively until C_i is disconnected. For the larger one of two yielded clusters, start the edge removal process again, and this process is repeated until the number of clusters reaches to k.

D. Repeated Bisection Clustering

As an application method of k-means clustering, Repeated bisection clustering has been studied. Generally, it is known as a faster and more accurate method than k-means clustering [28]. As stated in Algorithm 4, it recursively separates a graph into two clusters by assigning each node to a closer cluster. Note that the Step 2 in Algorithm 4, we select the nodes with minimum and maximum ID in C_i as two random nodes. Additionally, the distance of two nodes is defined as the shortest path length between the nodes.

VI. Simulations and Results

In this section, the simulation settings and results are presented. We have developed a simulator in Python and NetworkX to conduct our simulations and evaluate the clustering algorithms presented in Section V.

Algorithm 3 Betweenness Centrality Clustering.

Input: a graph G=(V, E), the number of domains to create k.

1: Start with an initial cluster C_i containing all nodes in G
2: **while** True **do**
3: **while** C_i is connected **do**
4: Compute $C_B(u, v)$ for all edges in C_i
5: Remove the edge with the highest $C_B(u, v)$
6: **end while**
7: **if** $|\mathbf{G^1}| - 2 = k$ **then**
8: Add two yielded clusters to $\mathbf{G^1}$
9: break
10: **else**
11: For two yielded clusters, add the smaller cluster to $\mathbf{G^1}$ and let the larger cluster be C_i.
12: **end if**
13: **end while**

Output: $\mathbf{G^1}$

Algorithm 4 Repeated Bisection Clustering.

Input: a graph G=(V, E), the number of domains to create k.

1: Start with C_i containing all nodes in G.
2: **while** True **do**
3: Select two random nodes v_a and v_b from C_i and create clusters C_a and C_b containing v_a and v_b, respectively.
4: **for** each node in C_i **do**
5: Calculate the distance with v_a and v_b and assign the node to the closer cluster.
6: **end for**
7: **if** $|\mathbf{G^1}| - 2 = k$ **then**
8: Add C_a and C_b to $\mathbf{G^1}$
9: break
10: **else**
11: For C_a and C_b, add the smaller cluster to $\mathbf{G^1}$, and let the larger cluster be C_i
12: **end if**
13: **end while**

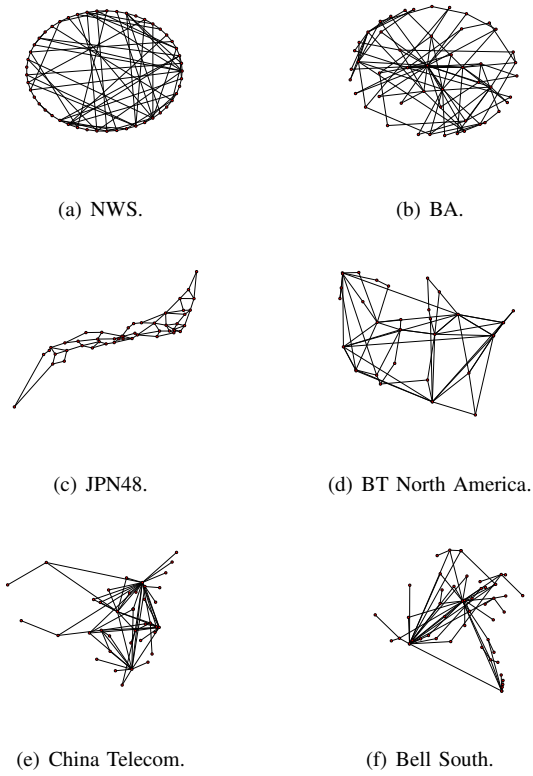
Output: $\mathbf{G^1}$

A. Network Models

Our simulation makes use of two types of random graphs and four kinds of real network models. As random graphs, Newman Watts Strogatz (NWS) and Barabasi Albert (BA) are used because they are flexible to adjust their sparseness and denseness for theoretical analyses. A NWS graph is formed by connecting random pairs of nodes with a certain probability after creating a ring over n nodes, which tends to be sparse [29]. In contrast, a BA graph is generated where nodes with higher degree have higher probability to be connected to a node during its generation process [30].

In addition to NWS ans BA graphs, the clustering algorithms are executed on the real network models: JPN48, BT North America, China Telecom, and Bell South provided in [31][32]. Figures 3(a) to 3(f) depict the network models used

in our simulations, and Table I describes the parameters of the network models, such as the number of nodes, edges, and average degree.



(a) NWS. (b) BA.

(c) JPN48. (d) BT North America.

(e) China Telecom. (f) Bell South.

Fig. 3. Network models for simulation.

TABLE I
SIZE OF NETWORK MODELS.

	NWS	BA	JPN48
# of nodes	50	50	48
# of edges	62	100	82
Average degree	2.48	3.88	3.41
	BT North America	China Telecom	Bell South
# of nodes	36	42	51
# of edges	76	66	66
Average degree	4.22	3.14	2.58

B. Evaluation Indicators

In our simulations, three different indicators are evaluated while the number of controller domains is varied.

1) **Load balance of controllers:** As mentioned in Section IV-C, the standard deviation of the switch weight in domains $\mathbf{G^1_\sigma}$ is evaluated to examine the load balance of controllers.

2) **Inter-control load:** The amount of topology information shared among controllers could be related to Inter-controller load since they need to synchronize network information they possess to keep a global view. As stated in Section III-C, the amount of shared topology information corresponds to the size of federation graph $|G^f|$ in the layered control plane. However, since the number of nodes in a federation graph $|V^f|$ is equivalent to the number of controller domains, $|G^f|$ varies depending on the number of the edges in a federation graph $|E^f|$, which is the number of inter-domain links.

3) **Reliability of controller domains:** It might be desirable for any switches to have multiple paths reaching to other switches within the same domain in case of failures; otherwise, the extra inter-controller communication might be required as addressed in Section III-C. Thus, as a indicator to evaluate the reliability of controller domains, the average number of edge disjoint paths for all pairs of nodes in a domain is compared.

Assuming that the number of edge disjoint paths between nodes u and v is given as ψ_{uv}, the average number of edge disjoint paths in a domain is denoted as

$$\Psi(G^l_i) = \frac{\sum_{u,v \in E^l_i} \psi_{uv}}{|V^l_i| C_2}. \quad (12)$$

Then, the average number of edge disjoint paths of a clustering as a whole is calculated as

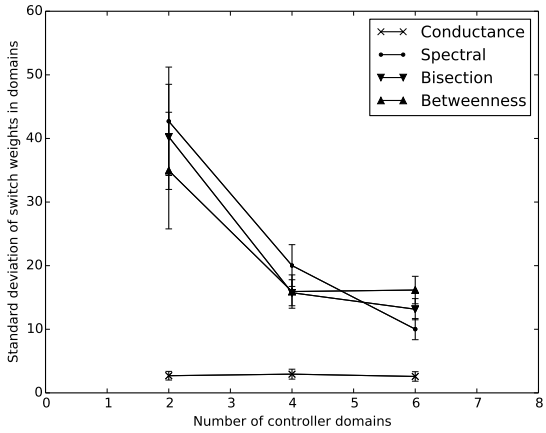
$$\overline{\Psi(\mathbf{G^1})} = \frac{\sum_{G^l_i \in \mathbf{G^1}} \Psi(G^l_i)}{k}. \quad (13)$$

C. Results and Discussions

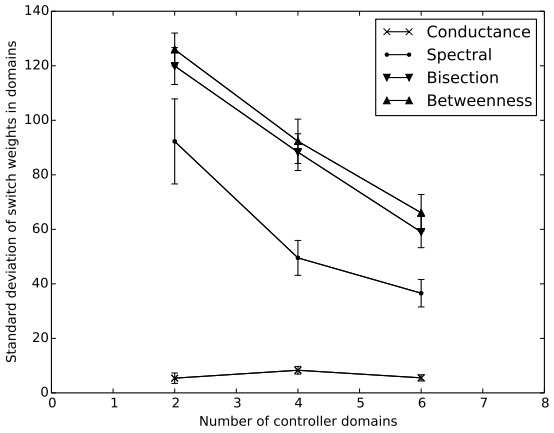
This section presents the simulation results and discussions. In our simulations, the number of controller domains is varied from 2 to 6 in all simulations, and each simulation is executed 20 times.

1) **Load balance of controllers:** Figures 4(a) to 4(f) illustrate the standard deviation of the number of the switch weight in domains representing the load balance of controllers. We can see from Figures 4(a) to 4(f) that Conductance clustering demonstrates better than any other algorithms for balancing the controller load. This would be because Conductance clustering generates a cluster until the number of nodes in the cluster reaches to the constraint of the number of nodes in a cluster, which is set as the number of nodes in an original graph divided by the number of controller domains. As a consequence, most clusters include almost the same number of nodes as the constraint even though last-produced cluster may contain less number of nodes compared with others. This depends on whether the number of nodes in an original graph can be divisible by the number of controller domains.

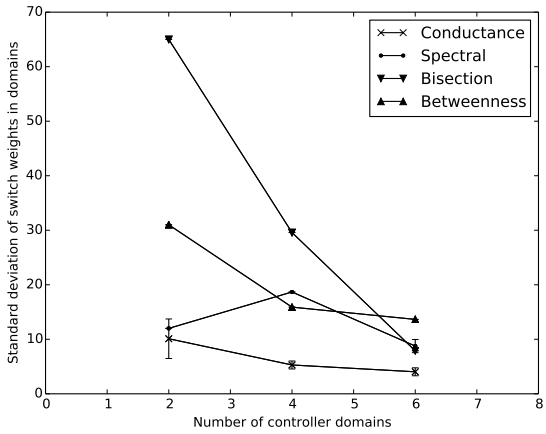
2) **Inter-control load:** Figures 5(a) to 5(f) indicate the number of inter-domain links implying the inter-control load. Those results show that Spectral clustering could generate clusters with less number of inter-domain links on NWS, JPN48, BT North America, and Bell South. On the other hand, Betweenness centrality clustering performs better on BA and China Telecom. As Spectral clustering separates a graph based on graph connectivity, it generally produces dense clusters and sparse relations among them, which yields less number of inter-domain links. However, the reason why Betweenness centrality clustering works better on BA and China Telecom would be because of their structural feature. Both models include several hub nodes, and the edges connecting such nodes



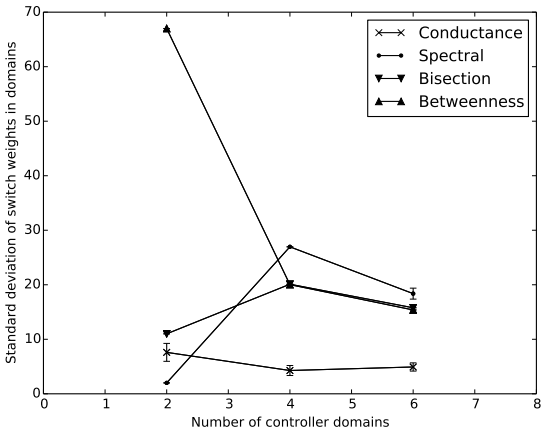
(a) NWS.



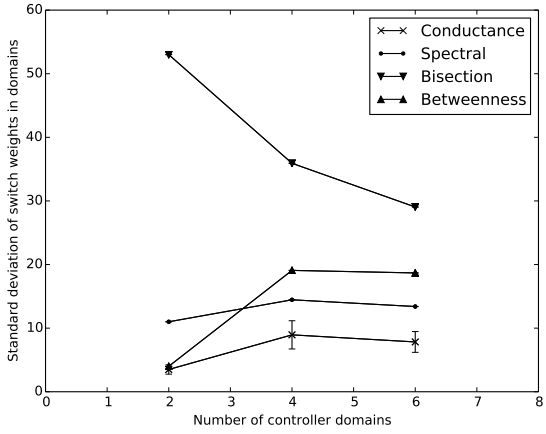
(b) BA.



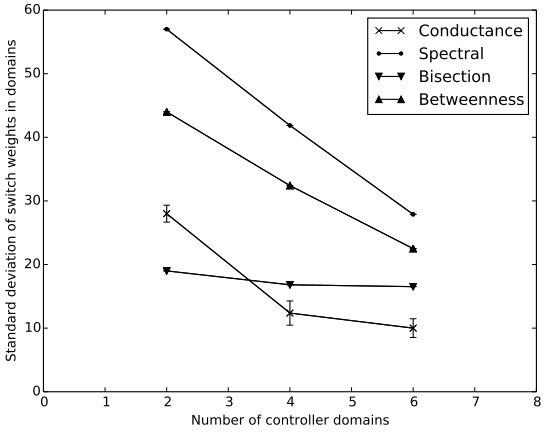
(c) JPN48.



(d) BT North America.

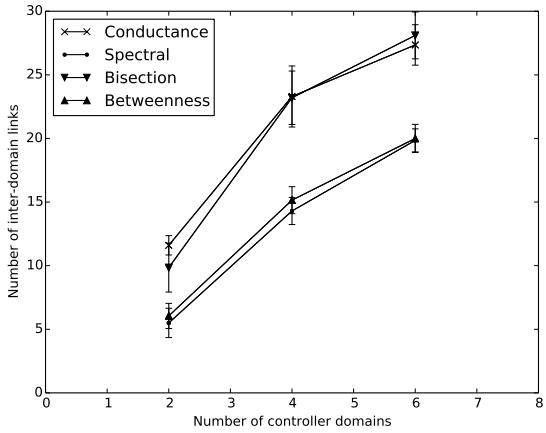


(e) China Telecom.

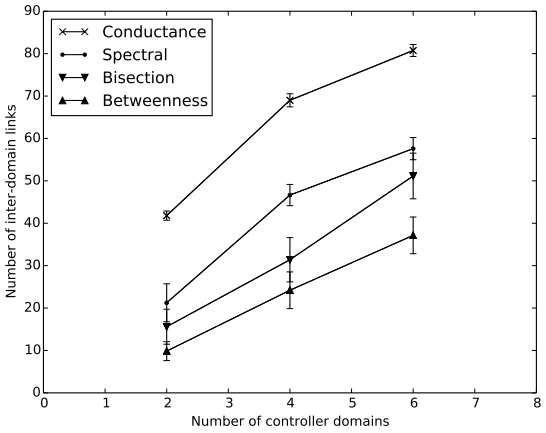


(f) Bell South.

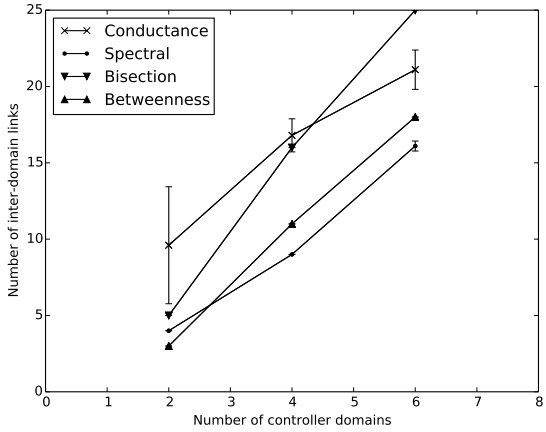
Fig. 4. Load balance of controllers.



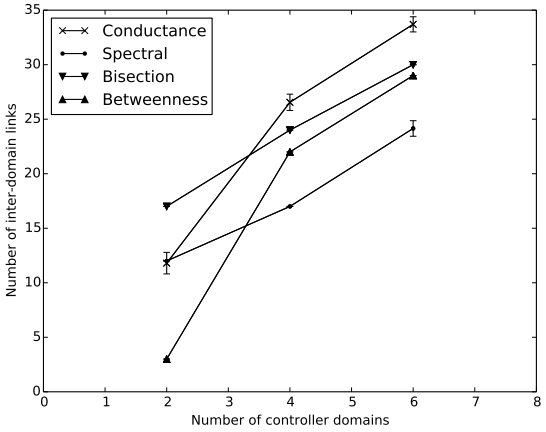
(a) NWS.



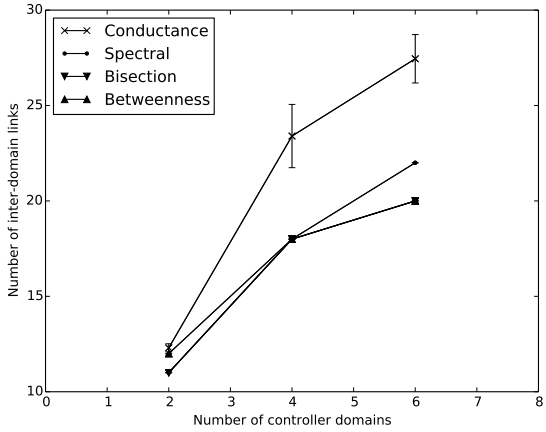
(b) BA.



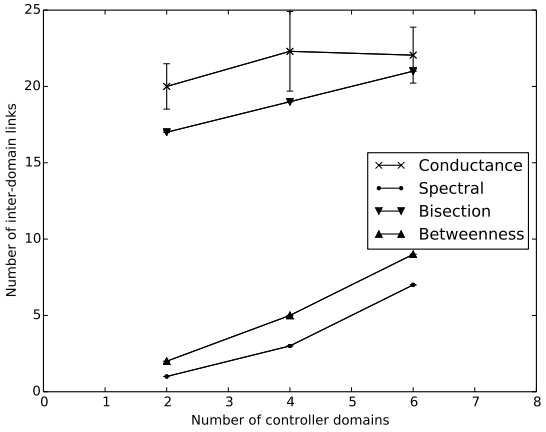
(c) JPN48.



(d) BT North America.

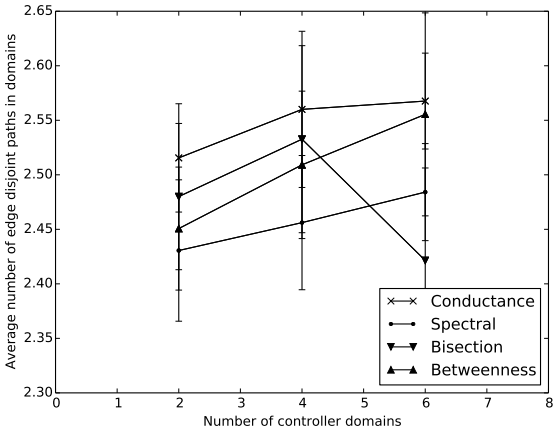


(e) China Telecom.

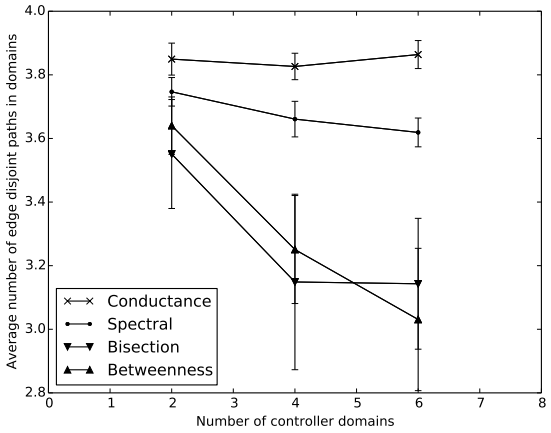


(f) Bell South.

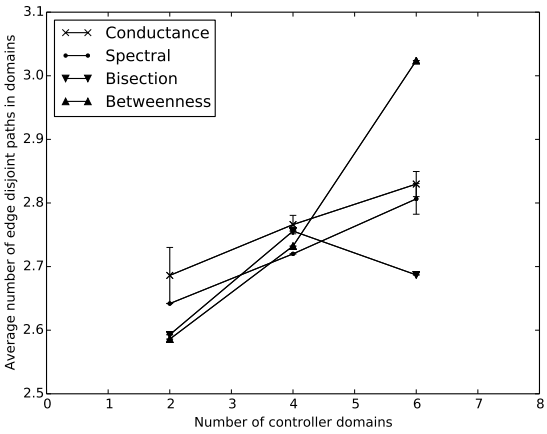
Fig. 5. Inter-control load.



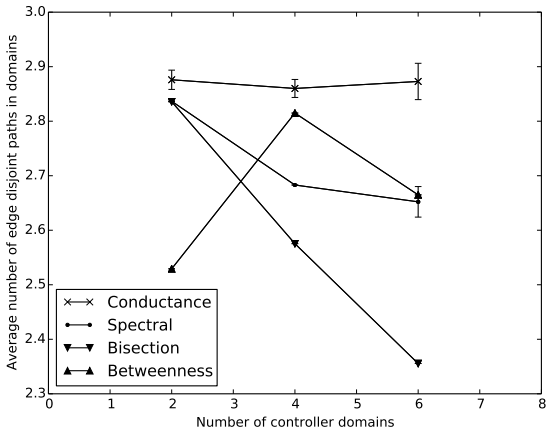
(a) NWS.



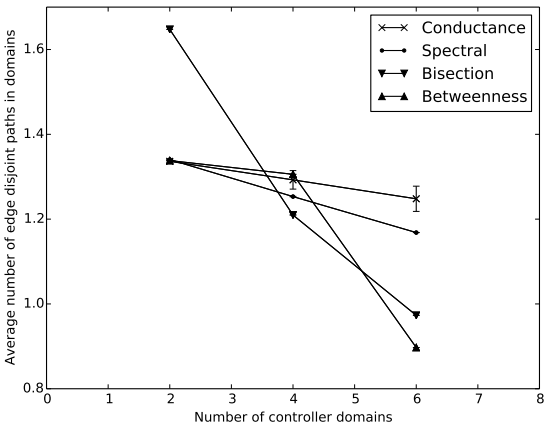
(b) BA.



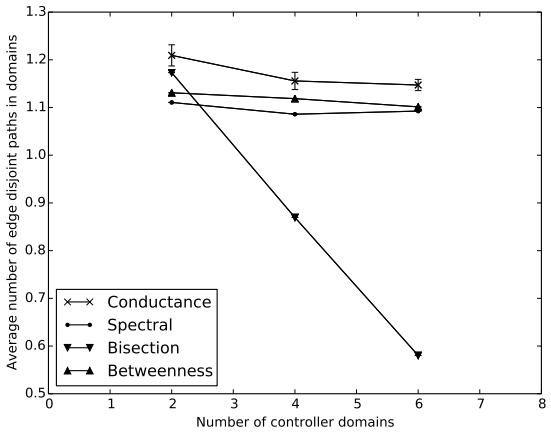
(c) JPN48.



(d) BT North America.



(e) China Telecom.



(f) Bell South.

Fig. 6. Reliability of domains.

would have high betweenness centrality. Thus, Betweenness centrality clustering can reveal the sub-structure of the graph by relatively a few edge removals. As a result, it may end up to produce the less number of inter-domain links.

3) **Reliability of domains:** Figures 6(a) to 6(f) depict the average number of edge disjoint paths in domains. As we can see from those results, Conductance clustering yields clusters with higher average number of edge disjoint paths on most of network models. Conductance clustering produces a cluster by choosing a node one by one so that the density of the cluster becomes high. As a result, it may increase the possibility to have many edge disjoint paths between nodes since the algorithm tries to contain many edges within each cluster.

In Figure 6(d), Betweenness centrality clustering results in the non-monotonic behavior when the number of controller domains is varied from 2 to 4. As mentioned in Section V-C, Betweenness centrality clustering partitions a graph by removing edges with high betweenness centrality until the graph is disconnected. Thus, it can be assumed that BT North America model might contain some dense substructures, and those might be coincidentally revealed by the edge removal process when 4 domains are created.

Moreover, in Figure 6(e), Repeated bisection clustering demonstrates the best result when two domains are created. However, this would be because it just separates a graph into two parts: extreme large cluster and small one. In fact, unbalanced clusters are created by the clustering method as seen in Figure 4(e). Hence, it could be said that the result is just originated from a single large cluster.

VII. Conclusion and Future Work

This paper has focused on the network partitioning for multi-domain SDN networks. By abstracting an SDN network as a graph, this paper approached to the issue based on graph clustering and formulated Network Partitioning Problem (NPP). Then, four clustering algorithms have been provided and evaluated with different SDN-related indicators, such as the balance load of controllers, inter-control load, and reliability of domains. The simulation results show that Conductance clustering could contribute to better-balanced load of controllers and higher reliability of controller domains while Spectral clustering demonstrates less inter-control load. Our future work will include a comparison of the clustering methods in terms of the controller placement. Furthermore, because the current simulations are only theoretical approach, an examination of different partitioning and its effects on network performances under realistic SDN scenarios are also left as our future work.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 26330120.7.

References

[1] A. Hidenobu, N. Junichi, and S. Norihiko, "Network partitioning problem to reduce shared information in openflow networks with multiple controllers," ICN 2015, 2015, pp. 250–255.

[2] S. Sezer et al., "Are we ready for SDN? implementation challenges for software-defined networks," Communications Magazine, IEEE, vol. 51, no. 7, July 2013, pp. 36–43.

[3] S. Kuklinski and P. Chemouil, "Network management challenges in software-defined networks," IEICE Transactions on Communications, vol. 97, no. 1, 2014, pp. 2–9.

[4] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," Communications Magazine, IEEE, vol. 51, no. 2, February 2013, pp. 136–141.

[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association, 2012, pp. 10–10.

[6] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 7–12.

[7] D. Kreutz et al., "Software-defined networking: A comprehensive survey," proceedings of the IEEE, vol. 103, no. 1, 2015, pp. 14–76.

[8] B. Pankaj et al., "Onos: towards an open, distributed sdn os," in Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1–6.

[9] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013, pp. 121–126.

[10] H. Xie, T. Tsou, D. Lopez, H. Yin, and V. Gurbani, "Use cases for alto with software defined networks," Working Draft, IETF Secretariat, Internet-Draft draft-xie-alto-sdn-extension-use-cases-01. txt, 2012.

[11] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in Teletraffic Congress (ITC), 2013 25th International. IEEE, 2013, pp. 1–9.

[12] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks." in OSDI, vol. 10, 2010, pp. 1–6.

[13] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 19–24.

[14] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The sdn controller placement problem for wan," in Communications in China (ICCC), 2014 IEEE/CIC International Conference on. IEEE, 2014, pp. 220–224.

[15] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On multi-controller placement optimization in software defined networking-based wans," ICN 2015, 2015, p. 273.

[16] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards sdn," in Communications (ICC), 2015 IEEE International Conference on. IEEE, 2015, pp. 369–374.

[17] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," Communications, China, vol. 11, no. 2, 2014, pp. 38–54.

[18] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving sdn survivability," in Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 1909–1915.

[19] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," Journal of the ACM (JACM), vol. 51, no. 3, 2004, pp. 497–515.

[20] S. E. Schaeffer, "Graph clustering," Computer Science Review, vol. 1, no. 1, 2007, pp. 27–64.

[21] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," 2014.

[22] U. Brandes and T. Erlebach, "Network analysis." Springer Berlin Heidelberg, 2005.

[23] F. Jordan and F. Bach, "Learning spectral clustering," Adv. Neural Inf. Process. Syst, vol. 16, 2004, pp. 305–312.

[24] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, vol. 17, no. 4, 2007, pp. 395–416.

[25] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," Applied statistics, 1979, pp. 100–108.
[26] U. Brandes, "A faster algorithm for betweenness centrality*," Journal of Mathematical Sociology, vol. 25, no. 2, 2001, pp. 163–177.
[27] M. Girvan and M. E. Newman, "Community structure in social and biological networks," Proceedings of the national academy of sciences, vol. 99, no. 12, 2002, pp. 7821–7826.
[28] S. M. Savaresi and D. L. Boley, "On the performance of bisecting k-means and pddp." in SDM. SIAM, 2001, pp. 1–14.
[29] M. E. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," Physics Letters A, vol. 263, no. 4, 1999, pp. 341–346.
[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," science, vol. 286, no. 5439, 1999, pp. 509–512.
[31] T. Sakano et al., "A study on a photonic network model based on the regional characteristics of japan (in japanese," 2013.
[32] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," Selected Areas in Communications, IEEE Journal on, vol. 29, no. 9, 2011, pp. 1765–1775.

A Passive Traffic Algorithm for Detecting Unavailable Periods in TCP Services

Iria Prieto, Mikel Izal, Eduardo Magaña and Daniel Morato

Public University of Navarre
Navarre, Spain
Email: iria.prieto, mikel.izal, eduardo.magana, daniel.morato @unavarra.com

***Abstract*—This paper presents a simple passive algorithm to monitor service availability. The algorithm is based on packet counting over a passive traffic trace of a population of clients accessing servers of interest. The major advantage of the algorithm is that it is passive and thus not invasive while usual monitor systems that can be found on Internet are active probing agents. The proposed system does not communicate to actual servers. It is easy to build as an online monitoring system with no big constraints in software or hardware. It does not relay on a distributed number of network placements for probing agents but works on a single network observing point near network edge. Initial proof of work of the algorithm is presented by studying the influence of different kinds of disruptions on packet level traffic and analyzing unavailability problems for popular servers at an academic network at Public University of Navarre.**

Keywords–Availability service; network; traffic

I. INTRODUCTION

As networks constantly evolve, network application servers are improved in software and hardware in order to cope with the growth of client's demand. In spite of this rapid development, sometimes, clients can not gain access to the servers due to communication problems or server saturation, due to flash crowd demands, human errors, updates, routing failures, etc.

Nowadays, even few minutes unavailability can be critical. For an enterprise offering products to clients through a web server, an interruption of this service means loss sales. Another example, which shows the threat of service interruption is the use of an antivirus update server. In case of banks, or other organization where security is a priority, an interruption of the update server entails possible infection problems.

As a result of the importance of being online all the time, several monitoring systems have been developed along the time. The target of these system is to detect as soon as possible when a network was experiencing problems. Typically, these kinds of systems are based on active probing. The main disadvantage of active probing is that sometimes, depending on the network condition, it will not be able to be applied. In high-loaded networks the methods of active probing, although it will try to be as simple as possible, is not always desired since an slight increment of the traffic may causes more packet delays and losses.

Nowadays, the vast majority of the services offered on Internet use as the application layer the Transport Control Protocol (TCP). The target of this work is to consider a passive method to detect service disruptions based on the study of the TCP packets. In [1] the algorithm was introduced showing the results for real services. In this work, the study has been extended to analyze how the unavailable periods caused by different problems can be distinguished using the proposal algorithm.

The paper is organized as follows. First of all, Section II describes the State of the art about the problem to face up. After that, the algorithm and configuration parameters are introduced in Section III. Section IV analyzes how the nature of the disruptions can affect the observed traffic and how it will be detected by applying the algorithm on the captured traffic. Section V describes the network scenario used to check the proposed algorithm. Section VI presents the results, comparing it to active detection of popular public services. Finally, Sections VII and VIII present conclusions and future work.

II. STATE OF THE ART

In order to detect when the clients of a network are not being able to successfully use a server application, a wide range of monitoring clients, such as Nagios [2], Zabbix [3], Cacti [4], Munin [5], have been developed. These systems warn the network administrator that a given server of interest is unavailable. These kind of systems work based on active probes, such as ICMP (Internet Control Message Protocol) ping or automatically requesting a server web page in case of monitoring HTTP (Hypertext Transfer Protocol) server. They are required to be installed and configured in monitoring client machines or at the server.

In cases where problems need to be detected at different client networks, at least one client has to be installed on each network. Otherwise, some problems will not be detected, like cases of routing problems in the path from clients to the servers of interest, if the monitoring client may use other route to reach the server.

As it is shown by Liu et al. [6], depending on the location of the system resources the application will achieve more effectiveness. Therefore, depending on where our monitoring clients will be located we would have only the vision of this location. Also, checking the configuration of these monitoring clients can be a problem for multi-tier system where the number of them will be high. In the literature, some papers explore how to face up testing the configuration in these scenes, [7]. Another problem of taking active measurement across an entire network is that for wide area networks it will not be scalable and some paths should be chosen and the rest of statistics inferred through predictive algorithms [8].

On the other hand, active probing can be a problem in high loaded systems or when monitoring third party servers, which may not react well to external continuous requests. Nowadays,

more and more enterprises rely on public services on Internet that would need to be monitored. In these cases, firewalls and intrusion detectors may deny probes or even ban future normal requests as response to continuous monitoring.

Configuring and using these kinds of distributed monitoring systems is not trivial as shown by different studies on how to approach the problem of monitoring for distributed programs [9]–[15].

Another disadvantage of active availability monitoring comes from cases where the clients access servers through proxy-caches. In that case, the monitoring client may be requesting a webpage from the server and receiving a response just because it is cached at the proxy system even if the final server is unreachable or has some problem. Thus, the active measurement does not actually check for server availability and other clients in different networks or served by different proxies may be experiencing access problems for the same server. In these cases the system would not detect the problem until the timeout of the cached object. This situation can be addressed by proxy configuration (may not be an option depending on proxy ownership) or crafting requests so they are not cached.

In some cases, due to misconfiguration or network issues, the monitoring client may experience problems to reach the server while actual client access is working, thus giving rise to false positive alerts to the network administrator. The cause of this failures may be things as memory problems or CPU or network overload of the monitoring client. This is often due to the fact that the same agent is probing a large number of servers. Therefore, the dimensioning of these clients has to be considered carefully.

Another issue to consider is the reaction time of the monitoring system. The minimum and maximum acceptable time for problem detection has to be decided. Longer times imply slower reaction, smaller times may generate higher overhead and interference to normal clients.

Currently, the majority of cloud services available on the Internet offer services over TCP protocol for communications with clients [16], [17]. It has been observed that some servers, due to overload, start refusing new TCP connections by answering with RST packets to clients for some time. In many cases the observed time of these kind of events is on the order of seconds, but usually less than half a minute. After this event the server recovers its normal behaviour and accepts again new clients. As stated before, even if it only lasts for seconds this problem may be critical for some businesses, causing user complaints and bad server reputation.

There have been proposals to cope with the downsides of active monitoring. Schatzmann et al. [18] proposed a method to detect temporary unreachability based on flow-level analysis by capturing traces from different routes. Although their method was able to work online the main disadvantage was the need to monitor in different points of a network. Besides, it should be taken into account that the setup of these kinds of measurements is not an easy task [19].

Others works have compared the advantages and disadvantages of using active probing or passive monitoring to service discovery [20]. This work was based on service discoveries and not in detecting disruptions so for passive traffic only observed the connection establishments and abrupt finalisation.

The goal in this work is the development of a simple online disruption detection method for TCP servers. It has been noticed how the disruptions affect packet level network traffic. This analysis of packet level traffic allows to distinguish when clients are having problems with one service. The proposed method avoids active measurement and works just by passive observing network traffic. This method is based on simple packet level counting such as the number of RST and data packet received. It does not require large amounts of memory or CPU power and it is able to detect problems for clients in different networks and for different services without using distributed agents. It will be shown that it is able to detect micro-access-failures with a configurable granularity in the reaction time.

III. Proposed algorithm

As stated in the introduction, the method is based on passive traffic capture. By capturing traffic close to the clients in a given network it will detect when some services will not be available to this community (in this work, the sample community will be the clients at Public University of Navarre network). The main target of the proposed algorithm is to find when a service disruption event has occurred, that means that the clients on the monitored network can not successfully use the service. The server may be down or may just be unreachable from this point due to network or some other problem. In any case this local unavailability is what the network administrator wants to detect more than the global server state. The objective is to detect availability problems, including the case where clients are able to reach the servers but not to use their services. To achieve this, a simple algorithm has been proposed, which does not require big hardware or software constraints.

The flow of traffic from the clients to the servers of interest is captured and some simple counters are evaluated every fixed time interval. The counters used are the number of data packets and reset packets sent by the full group of clients and target servers seen during a given (i.e., 5 seconds) time interval. Reset packets are TCP protocol packets with RST flag activated. They are used by a TCP endpoint to reject incoming connections and also whenever an abnormal packet is received by a TCP endpoint, to signal to the other side that it should abort the connection. The algorithm bases on the fact that a server sending just TCP RST packets and not any other valid packet to a group of clients during even a small period of time is an indicator of unavailability. Although sometimes it has been observed that the servers finish their connections in an unexpected way such as, sending RST packets to the clients after a client has sent a FIN packet, the algorithm will not show false positives since it will have a high probability that another client will be sending or receiving data packets in the same period. The mechanism consists on dividing time in fixed sized intervals. On every interval the number of packets seen from clients and servers are considered and related to previous interval. When a client sends packets to servers, which do not send anything back to it, a server issue is suspected.

If in subsequent seconds the servers keep silent but send reset packets the servers are confirmed as not working. Also, if the client keep sending packets and the servers keep silent it is confirmed as not working. The previous identification idea is built with two simple filters for every interval. On

each time interval, counters for clients and servers are updated in order to describe the situations explained before. On the side of the client, the counter is the number of packets sent to the servers, regardless if they are data packets or not, $packet\_cli$. On the other hand, on the side of the server, two counters are taken into account: The number of data bytes sent, $bytes\_servers$, and the number of packets with the reset flag activated, $reset\_packets$.

If during a given interval the counters show the client was sending packets but the servers did not send any data packet (even they may send reset packets) the result of the first filter for that time slot is 1. Also, the result is 1 when there are no packet sent by the client and the server only sends RST packets. That indicates the server is not answering requests. The second filter would be 1 whenever the result of the first filter of the interval being analazying is 1 and the result of the first filter of the previous interval was also 1. The process can be easily explained through two membership functions, like the ones used in fuzzy logic [21], which are applied in each period. Firstly, the used variables are defined:

- x= Number of client packets sent in an interval.
- y= Server Bytes sent by the servers in an interval.
- z= Number of RST packets sent by the servers in an interval.
- i = i^{th} Interval to be analazyed.
- $\psi_i(x, y, z)$= First pass of the compound filter applied in each interval i.
- $\varphi_i(\psi_i, \psi_{i-1})$= Second pass of the compound filter applied in each interval i, it takes into account the result of the first pass.

The two membership functions are described in equation 1.

$$\psi_i(x,y,z) = \begin{cases} 1 & if \quad ((x>0) \quad and \quad (y=0)) \quad or \\ & ((x=0) \quad and \quad (y=0) \quad and \\ & \quad (z>0)) \\ 0 & Otherwise \end{cases}$$

$$\varphi_i(\psi_i, \psi_{i-1}) = \begin{cases} 1 & if \quad (\psi_i = 1) \quad and \quad (\psi_{i-1} = 1) \\ 0 & Otherwise \end{cases} \tag{1}$$

Each period is labeled with the result of applying the two membership functions, $(\psi_i(x, y, z), \varphi_i(\psi_i, \psi_{i-1}))$. When both results are 1 an availability problem is considered for the duration of both intervals. We define the unavailability period since the first second of the interval labeled as $(1, 1)$ until the next interval labeled as $(0, 0)$. An example of the algorithm operation is shown in Table I.

In the second interval of Table I, there was one packet sent by a client, but there was no data sent to him by servers so the first flag is 1 and the second one is 0 because it was the first suspected interval. After this first interval, the servers, which belong to Hotmail service, sent 8 packets being all of them TCP RST packets. As there were only reset packet we label this second interval as $(1, 1)$. During the next 5 seconds the servers seem to have recovered because data packets from servers are seen again.

The example is a real case disruption interval detected for Hotmail server at the scenario. During that interval only reset packets were captured from servers and the packet trace was examined to show that servers were closing connections that had been inactive for more than 30 seconds.

These resets were not a response to any observed packet, so it seems reasonable that the server was experiencing problems and thus this is the kind of event the algorithm addresses. The main parameter of the algorithm is the time interval duration, that can be chosen by the network administrator depending on the desired reaction time. Smaller values will increase resolution and will detect microfailures but will also increase false positives.

From our experience, values between 5 and 15 seconds are recommended.

IV. Analazying the effect of the disruptions on the traffic observed

As it has been explained in the introduction, different problems can affect the client network. In this section, the effect of some of the most typical connection problems and how these problems are revealed on the traffic at packet level are studied.

For this purpose a testbed was set up, since the study of the nature of the problems in a real network would mean to have documented why each unavailable interval happened. In cases where the network suffered a problem close to the sniffer, those intervals could be labeled, but if the problems were outside our university network it is not possible to know exactly the cause.

Using the testbed unavailable periods between clients and servers were created intentionally. The duration of these conditions can be decided at first and, after some minutes where the clients will not able to gain normal access to the service on the server, the problem will be solved. These periods are labeled and compared to the values obtained by applying the proposed algorithm.

The following subsections described de testbed scenario, the effects of the disruptions on the traffic captured and how the algorithm detects these intervals.

A. Testbed scenario

The testbed consists of two client networks where several agents continuosly request a webpage to the server. Both networks are connected to the servers networks by two routers as seen in Figure 1. These routers perform also the role of sniffers. All the scenario was emulated using a completely virtual environment using Virtual Box. All the virtual networks operate at 10Mbps with a loss rate of 1%. The purpose of the loss rate is to make a more realistic environment. The whole scenario is shown in Figure 1.

As the amount of traffic can influence the time when the unavailable periods are detected by the algorithm, the requests are made following exponentially distributed interarrival times whose average load are different on both networks. The network 1 had an average load higher than the network 2, 8 Mbps and 3 Mbps, respectively (Table II).

One of the most frequent problems that networks suffer is due to link disconnections. Sometimes some link in the path between the client network and the server falls down by hardware problems on some interface or maybe the link

TABLE I. EXAMPLE OF THE DOUBLE CHECK ALGORITHM DEVELOPED FOR A INTERVAL OF THE DAY 2013/11/8 AND HOTMAIL SERVERS

Start	End	Bytes Serv (x)	RST Serv (z)	Packet Cli (y)	ψ	φ
9:24:55	9:25:00	7016	0	14473	0	0
9:25:00	9:25:05	0	0	1	1	0
9:25:05	9:25:10	0	8	0	1	1
9:25:10	9:25:15	1699	1	3288	0	0

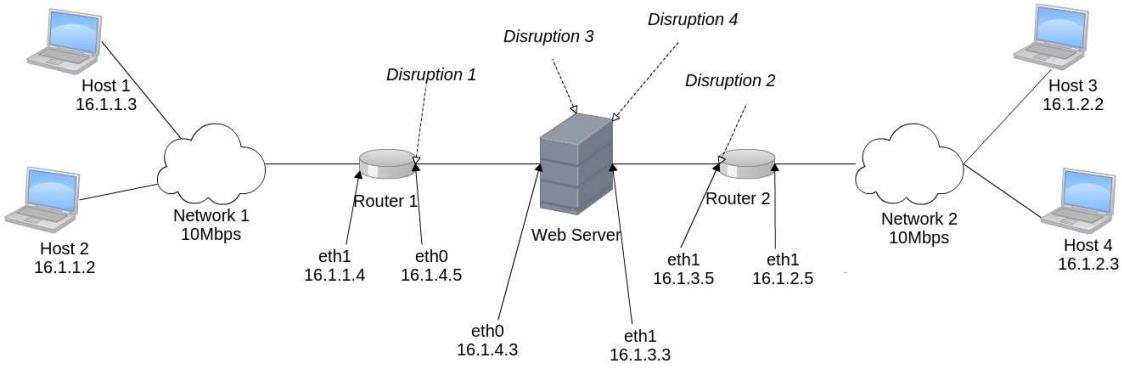


Figure 1. Testbed used to study the effect of different disruptions

TABLE II. CHARACTERISTIC OF THE NETWORKS

	Network 1	Network 2
Clients	2	2
Mean load	8 Mbps	3 Mbps
Losses (%)	1%	1%

is not disconnect but any packet is observed because the route is being changed. This unreachability situation is emulated in the testbed by switching off the outgoing interfaces on the respectively routers, at different times. After a short period the interfaces were switched on again. We will refer to these disruptions as Disruption 1, when the link on the router 1 was switched down but the rest were working properly, and Disruption 2 when the link on the second router was switched down.

Another usual service problem is due to servers being rebooted. Even if a service is being offered to clients based on a pool of servers, a particular client network can be affected by a single reboot on one server while the traffic of the clients is not addressed by another server. The reboot can be provoked by software updates, which require the reboot, by a hardware problem or simply by bad working. In the testbed, as a unique server was being emulating this error affected to both client networks. The reboot was made by a command and it is the time that took to recover the normal running wrote down; we will refer to this interruption as Disruption 3.

Finally, another common problem is that suddenly the actual service software falls down or does not respond. These kind of problems are caused, for example, by misconfigurations or changes on the services, for instance a port change, or by software problems. They are different from the previous one because in these cases the packets sent by the clients will be received on the server but it will not answer their requests. This problem was emulated by stopping the apache service on the server during some minutes. It was labeled as Disruption 4 and again it affected to both client networks.

All the disruptions described are emulated and it can be observed in the Table III. The estimated disruption time is the time, which was written down when the disruption started while, the estimated recovery time is the time that the service was available again. The estimated times between the disruptions and its recovery are approximated, in spite that these times were carefully taken, because they were written down by a human observer.

B. *Effects on the passive captured traffic*

The different problems emulated will not have the same effect on the observed traffic. In fact, in the cases of the disruptions 1 to 3, the packets sent by clients will not be able to reach the server, so it will not be received any TCP server packets. In the last case the clients will reach the server but they will not establish the TCP connections, so the server will likely answer with TCP Reset packets.

The traffic is analysed, at packet level, for each disruption near the time of the disruption was writte down. As the algorithm works on the TCP traffic, only this traffic was considered. In the two first cases, disruption 1 and 2, the clients suddenly did not receive any more packets sent from the server. In addition, it could be observed a lot of client connection attempts without any answer packet of the server. The attempts are showed by SYN packets sent from the clients. Both networks behaved identical for the disruptions 1 and 2 respectively. For this reason, only some packets, which belonged to network 1 are shown:

```
17:56:31 IP 16.1.1.2.37011 > 16.1.4.3.80: Flags [S],
17:56:33 IP 16.1.1.3.41460 > 16.1.4.3.80: Flags [S],
17:56:37 IP 16.1.1.2.37012 > 16.1.4.3.80: Flags [S],
17:56:41 IP 16.1.1.2.37013 > 16.1.4.3.80: Flags [S],
17:56:41 IP 16.1.1.3.41461 > 16.1.4.3.80: Flags [S],
17:56:42 IP 16.1.1.2.37014 > 16.1.4.3.80: Flags [S],
17:56:44IP 16.1.1.3.41462 > 16.1.4.3.80: Flags [S],
```

The reboot of a server may affects the traffic observed at packet level. Again, the traffic is studied close to the time,

TABLE III. PROGRAMMED DISRUPTIONS

Name	Type	Network 1	Network 2	Estimated disruption time	Estimated recovery time
Disruption 1	Network problem	Affected	Not Affected	17:45	17:47
Disruption 2	Network problem	Not Affected	Affected	17:56	17:59
Disruption 3	Reboot server	Affected	Affected	18:10	18:13
Disruption 4	Service not working	Affected	Affected	10:03	10:05

which was marked for the disruption 3 in the two networks. As in the case before, the clients suddenly stopped receiving packets from the server and the new requests did not find any answer from the server. During these intervals a lot of SYN packets were observed:

```
18:10:14 IP 16.1.1.2.37137 > 16.1.4.3.80: Flags [S],
18:10:15 IP 16.1.1.2.37137 > 16.1.4.3.80: Flags [S],
18:10:17 IP 16.1.1.2.37137 > 16.1.4.3.80: Flags [S],
18:10:21 IP 16.1.1.2.37138 > 16.1.4.3.80: Flags [S],
18:10:21 IP 16.1.1.2.37137 > 16.1.4.3.80: Flags [S],
18:10:21 IP 16.1.1.3.41681 > 16.1.4.3.80: Flags [S],
```

Observing the traffic at packet level of the previous disruption with the last one, an important difference can be observed. While in the previous cases the clients did not receive packets from the server, in the last one they were able to reach the server but as the service was not running this one answered them with Reset TCP packets. In the case of the last disruption, if we were probing the availability of the service using ping requests for instance, we would not realize of the problem since the server would answer. The effect on the traffic of the two networks was identical on both networks:

```
10:04:50 IP 16.1.4.3.80 > 16.1.1.3.59757: Flags [P.],
10:04:50 IP 16.1.1.3.59757 > 16.1.4.3.80: Flags [.],
10:04:50 IP 16.1.1.3.59757 > 16.1.4.3.80: Flags [R.],
10:04:55 IP 16.1.1.2.47925 > 16.1.4.3.80: Flags [S],
10:04:55 IP 16.1.4.3.80 > 16.1.1.2.47925: Flags [R.],
10:04:55 IP 16.1.1.3.59758 > 16.1.4.3.80: Flags [S],
10:04:55 IP 16.1.4.3.80 > 16.1.1.3.59758: Flags [R.],
10:04:57 IP 16.1.1.2.47926 > 16.1.4.3.80: Flags [S],
```

Therefore, once analysed all the cases, each of them will have little or any traffic towards the client from the server. The two functions used by the algorithm will adapt with the real scenarios since the server will not reply to clients or it will send reset packets until the problem is solved.

C. Detecting the unavailable periods

During the experiments shown on a real scenario, the time interval duration chosen was 10 seconds. The intention is that it is large enough to detect minute intervals with problems and at the same time, it is not too much small to rise false positives. Again, we used this value to define the durations of the intervals on the testbed. The target was to check if this amount of time was valid for the network with a high load and for the low one.

The unavailable periods showed by applying the algorithm to the sniffed traffic are shown in Figure 2. The periods corresponds quite accurately with the estimated time estimated (Table III). No false positive was showed up.

Comparing the full volume of traffic sent by the clients of each network with the server, again the difference between the different kind of the disruptions can be seen. For the disruptions where a link will be down for an interval of period, as the disruptions 1 and 2 any packet sent by the server will be observed but in contrast, a little amount of traffic sent by clients

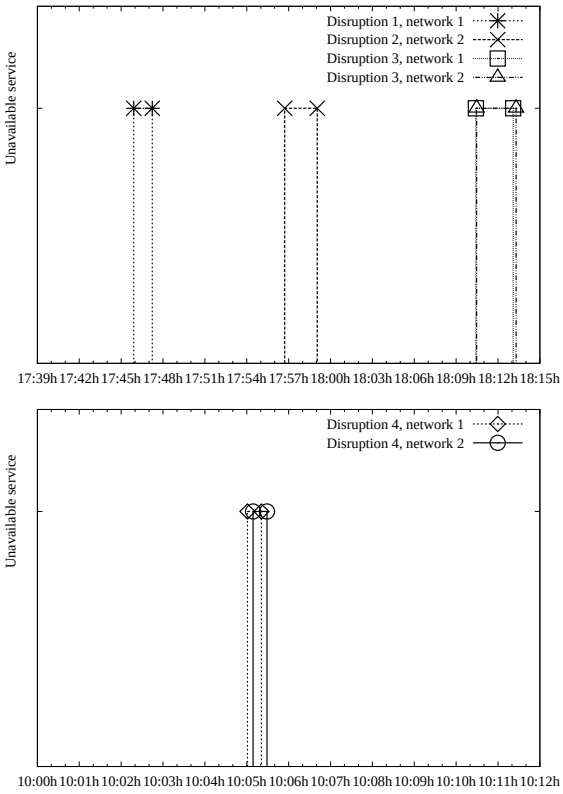


Figure 2. Unavailable periods detected on the testbed

trying to gain access to the server will be shown (Figure 3). In Figure 3, it is drawn the amount of Bps seen by the clients or servers addresses and the intervals of disruptions.

However, when the server is the one that suffers a problem, because it is being reboot or the service has fallen down, the effect on the traffic is quite different as it can be observed for the disruption 3 in Figure 2 and for the disruption 4 in Figure 4. While the first one the server will not send any packet, the last one will answer the requests with Reset packets. This means that in the last case it will observe traffic although it will be slightly lowest than before suffering the problem.

Another consequence of comparing the volume of traffic with the unavailable periods detected, is that in case of the second network the intervals are detected a little later than the network one, as it is shown in Figure 4. This is due to the clients on the second network do not request so frequently for the web page, so the problem is detected when they try to establish the connection with the server, which was later than the clients on the network 1. Similarly, the algorithm decides that the network 2 has recovered completely later than in the case of the network 1. On the second case, due to the clients

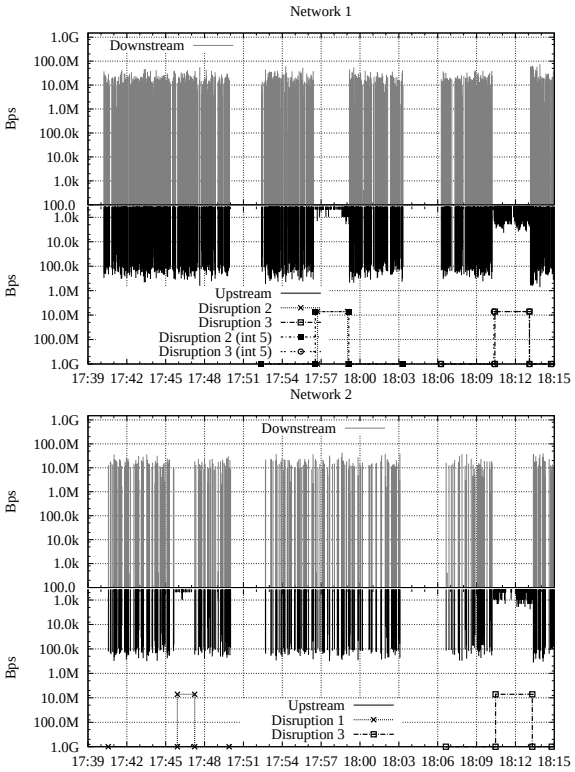


Figure 3. Bps for the use of Web service by the clients on the testbed at disruptions 1 to 3

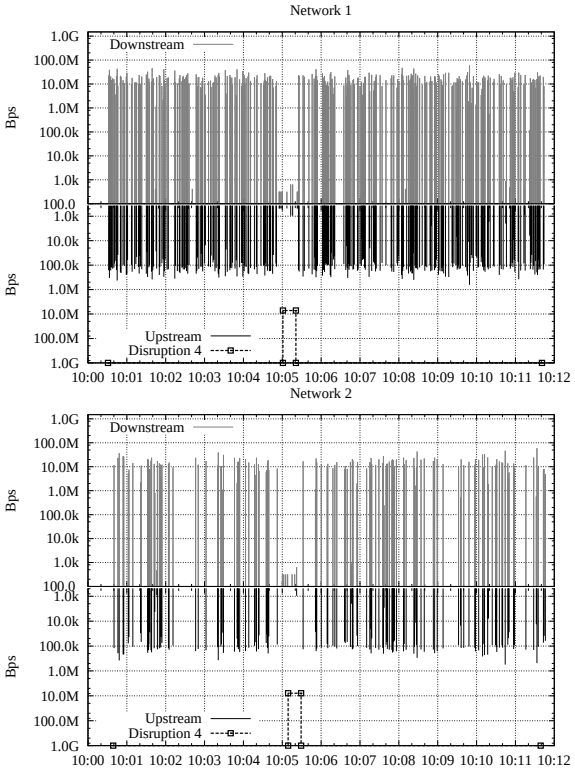


Figure 4. Bps for the use of Web service by the clients on the testbed at disruption 4

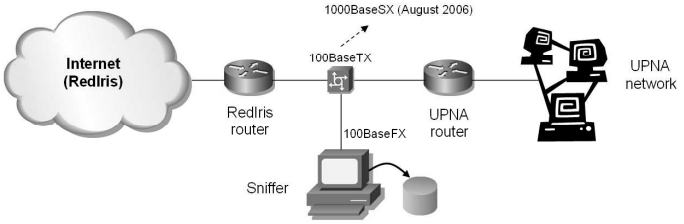


Figure 5. Traffic capturing from a University link

send less packets, the algorithm is not able to distinguish when the server is answering properly again. However, these difference between the unavailable periods detected for each network, are minimal, as can be seen in Table IV.

Another issue addressed in this section is the analysis of the impact of the time interval chosen in the algorithm. With a time interval of 10s it was checked as the results were quite accurate. Nevertheless the algorithm is applied using an interval smaller, exactly 5s. The main problem of using values for time intervals small is that the number of false positives could be increased since if the period is too small maybe the server had no time to reply at the same interval. However, in the testbed a period of 5 seconds was viable since the number of false positives was zero. The differences between using 10s and 5s were not big (Table IV). As the intervals were smaller the algorithm detected problems some seconds earlier. Moreover, the recovery times for each disruption were earlier detected also.

In spite of the fact that a value of 5 seconds behaves a little better in this scenario, in real scenarios where maybe the traffic is bursty or almost non existing at some hours, for instance at nights, we strongly recommend values of 10s instead of 5s. This value will decrease the possible numbers of false positives and still will be able to detect short intervals, in which clients were having problems to use the services properly.

V. NETWORK SCENARIO

The algorithm has been developed and tested, detecting availability problems of public internet servers for clients at Public University of Navarre. Captured data comes from author's research group infrastructure who has access to a sniffer with its own software between university main access and academic internet provider (Rediris) as seen in Figure 5. The group has an ongoing packet trace collection campaign since 2004 providing 1Gbps traces from the access of an academic community.

In this work, results are presented from captured data of the week of November 7th to 11th, 2013, checking the availability of popular servers at this community such as Facebook, Yahoo, BBC and Hotmail. In order to compare the algorithm against an active monitor (like Nagios [2]), a very basic probing system is implemented. The active monitor tests the availability of selected servers by requesting the site `favicon.ico` file. This file provides an icon to be displayed at browser window and is widely used by web servers. The program requests the favicon file every 5 seconds for every service considered in the experiment and thus provides a ground truth value of availability for comparison purposes.

The active requests are performed from a desktop computer at the university network. The number of servers probed is

TABLE IV. COMPARISON OF THE UNAVAILABLE PERIODS DETECTED APPLYING DIFFERENT TIME INTERVALS, 10S AND 5S

Network	Disruption	Estimated disruption period	Unavailable interval (t=10s)	Unavailable interval (t=5)
Network 1	2	17:56 - 17:59	17:56:33 - 17:59:03	17:56:23 - 17:59:03
	3	18:10 - 18:13	18:10:15 - 18:13:05	18:10:10 - 18:13:00
	4	10:03 - 10:05	10:04:51 - 10:04:51	10:04:46 - 10:05:16
Network 2	1	17:45 - 17:47	17:45:44 - 17:47:14	17:45:39 - 17:47:09
	3	18:10 - 18:13	18:10:18 - 18:13:18	18:10:13 - 18:13:13
	4	10:03 - 10:05	10:04:59 - 10:05:29	10:04:54 - 10:05:24

not very large so the probing computer is not loaded and no request failures can be attributed to machine overloading. The proposed passive algorithm operates on traces obtained at network edge as seen above. It is evaluated offline for the results of this work, but may be easily programmed as an online system.

As servers used are very popular, there are other sources of availability information that were considered. Several web pages provide down times and real time user complaints of public servers but usually this information has not enough time granularity to test less than ten minute disruption events.

VI. RESULTS

In this section, results of unavailability detection with a week trace of traffic are presented (November 7th to 11th, 2013). Public servers addressed are: "Yahoo", "Facebook", "BBC", "Hotmail" and also a local newspaper "Diario de Navarra", which are frequently visited by users at the University. Those servers, except the local newspaper, are also used by a large mass of users around the world and they are served by a pool of different IP addresses. They are probably distributed over large server farms or content distribution networks.

But even if those farms are probably designed to balance load and support peaks of demand, sometimes, the clients of the University are not able to reach these services.

Experiments with the basic active monitor that request `favicon.ico` file show the results in Table V for the servers under analysis. Figure 6 shows the events of unavailability with time. The service with more suspected intervals detected was Hotmail.

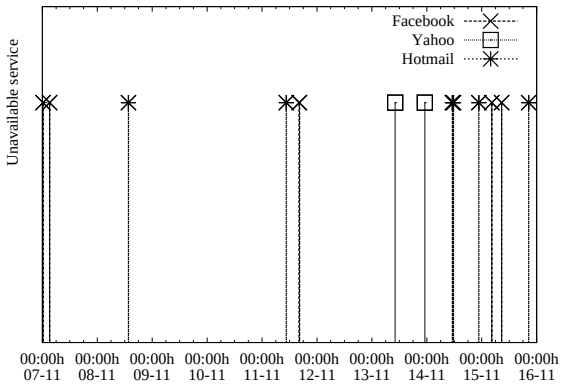


Figure 6. Events of time where the favicon was not be obtained

To test the proposed algorithm the packet trace of a full day is processed and the algorithm is applied on the traffic.

TABLE V. UNAVAILABLE SERVICE INTERVALS DETECTED BY REQUESTING THE FAVICON

Start	End	Day	Service
0:15:49	00:16:58	07/11/2013	Facebook
3:10:01	03:10:06	07/11/2013	Facebook
13:36:35	13:36:51	08/11/2013	Hotmail
16:08:12	16:25:40	11/11/2013	Facebook
10:34:59	10:35:21	11/11/2013	Hotmail
10:10:23	10:10:29	13/11/2013	Yahoo
23:08:21	23:08:27	13/11/2013	Yahoo
11:08:54	11:09:06	14/11/2013	Hotmail
11:39:18	11:39:36	14/11/2013	Hotmail
22:43:00	22:43:05	14/11/2013	Hotmail
8:40:31	08:40:44	15/11/2013	Facebook
20:30:03	20:30:13	15/11/2013	Hotmail
4:22:29	04:22:34	15/11/2013	Facebook

The rest of the results are for day 08/11/2013 although other days are similar.

First, the network traffic is filtered to select packets from the probing agent and selected servers of interest. Although this is not the target of this work, addresses of these servers have first to be identified. To solve this, the payload of packets is examined to search for these server names in HTTP requests.

Both methods active and proposed algorithm show some unavailability issues for the Hotmail service, see Figure 7. The plot shows the volume of traffic from client machine to Hotmail as well as the time events identified by the passive algorithm and active favicon requester. Both algorithms identified the same event. Packet level examination of the event showed a single connection, which suffered an unexpected reset from the server. The comparison also revealed that the time difference is due to the monitor client, which was not NTP synchronized as the passive sniffer is. This shows a point to take into account in a distributed monitoring system when monitor clients are distributed time synchronization plays a critical role. The passive sniffer has a unique clock source so the problem of synchronization is simplified.

Packet level analysis of previous event showed the dialog of the packets below. The *x.x.x.x* represents the IP of the client and the *y.y.y.y* the IP of a Hotmail server. After the connection is established, the client sent the request through a push packet of 176 bytes. Usually, after this packet was sent by the client the server answered with the `favicon.ico`. However, in this case the server sent an ACK packet without data and after some seconds, around 11, closed the connection sending a reset packet. This kind of behaviour is unexpected and during these seconds the client would have noticed a malfunction using the service.

```
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: S
13:36:02 IP y.y.y.y.http > x.x.x.x.59133: S
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: . ack 1
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: P 176
```

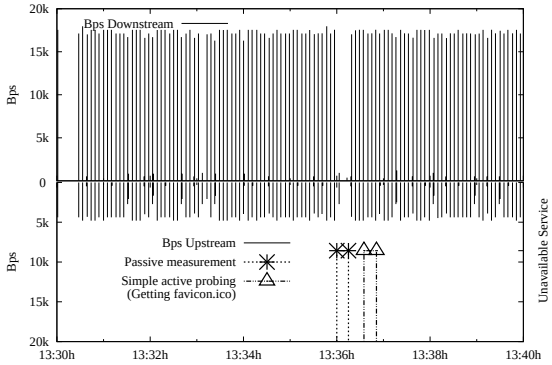


Figure 7. Intervals of time in, which the monitoring client had problems for day Nov 8th

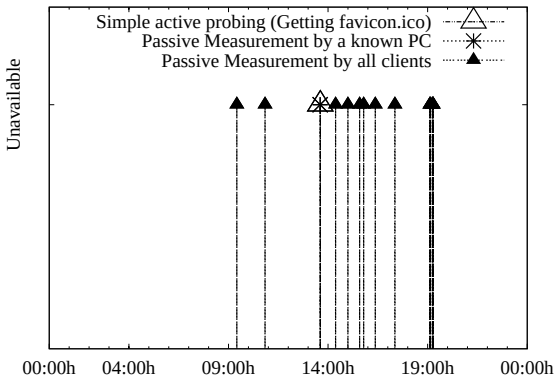


Figure 8. Comparison of the events of unavailable Hotmail service detected from request client for day Nov 8th

```
13:36:02 IP y.y.y.y.http > x.x.x.x.59133: . ack 177
13:36:13 IP y.y.y.y.http > x.x.x.x.59133: R 1 ack 177
```

Others cases of non-typical reset packets were also observed in the intervals of unavailability studied. In many cases, before a server went down it did not answer to the clients, and after some time it started to send them reset packets to clients since they did not reconignize the previous established connections.

A. Comparison between active probing vs passive analysis unavailability detection method

The total traffic from all the clients using services that previously have been identified to have unavailability periods is analysed. The objective is to distinguish the periods of time where all the users experience service access problems of the periods of time of isolated problems for individual clients.

To achieve this for each service, all the requested servers are joined together to study if in some period the clients were active but the servers were not working properly. The proposed algorithm is applied to the aggregated network traffic. The algorithm is configured using the IP addresses of all the servers as an unique service to be monitored and a time interval duration of 5 seconds. The unavailability events detected are shown in Figure 8. Interestingly, there are more unavailable periods detected that way than the issues detected by using the favicon requester alone.

The previous event, which was observed through the favicon.ico requests and observing the traffic for a single client who requests the favicon.ico (Figure 7), now is not labeled as problematic because at the same time other clients were able to use Hotmail. This interval was a problem of one server giving service to an individual client but it was not a problem of availability for the observed server since other clients were using the same service (other IP addresses of the same service). Thus, this is revealed as a false positive warning that shows the risk of using only the monitoring client as a method to detect service failures.

But this experiment shows other more important fact. By using the service as an aggregation of individual IP address of servers we are able to identify some unavailability intervals of a few seconds where the clients were suffering access problems but were not detected by active monitoring clients. These

TABLE VI. UNAVAILABLE HOTMAIL SERVICE INTERVALS

Start	End
09:25:05	09:25:15
10:50:00	10:50:10
14:22:40	14:22:50
14:59:40	14:59:50
15:35:00	15:35:10
15:47:15	15:47:25
16:22:05	16:22:15
17:21:10	17:21:30
19:06:50	19:07:00
19:07:20	19:07:35
19:13:35	19:14:05
19:16:10	19:16:30

periods were not observed by the monitoring client because the favicon.ico was served by a proxy cache. Table VI shows all the final disruption events detected.

These periods correspond to the sending of unexpected resets by the severs to the clients. The study of the traffic did not reveal any previously wrong behaviour of the clients, which could provoke the send of resets packets by the server. During this seconds, suddenly one or more servers decide to abort the established connections with one or more different clients. As the duration of the intervals were short, these were not actually critical disruptions since the next connections were established. In case that this kind of periods had to be ignored it may be done by just increasing the time interval duration, for example, to 10 seconds.

TABLE VII. UNAVAILABLE HOTMAIL SERVICE INTERVALS

Start	End
15:34:50	15:35:10
19:13:40	19:14:10

Table VII shows events detected from the same traffic by using an interval duration of 10 seconds. Two cases detected correspond to two intervals of 20 and 30 seconds. During this time there were only reset packets sent from the servers to clients, which have previously completed a connection establishment. Other intervals of 10 seconds are not detected since as the service recovered faster the reset packets sent in order to abort client connections felt inside the same interval

as the data packets sent by the servers once that they had recovered. Also, the intervals may not coincide exactly due to interval and event synchronization. The maximum error will be given by the minimum interval of time considered. For example, in the examples presented in this paper, the time of the disruption would be more or less 5 seconds since the interval is said, or 10 seconds when this is the used time interval.

We have checked also the rest of services whose some intervals were detected as unavailable by the monitoring client. The study of the traffic did not reveal any period with problems, there were not any interval of time where the server did not answer to the clients. The periods showed by the monitoring client were due to problems of the own client with the proxy cache or a particular server but not with the service.

B. Traffic profiling of the requested services

As a sanity check the full volume of traffic from the scenery network to the servers is observed to check that the amount of traffic was significant. Traffic for the 8th of November to Hotmail service is shown in Figure 9. The first image in the figure shows the whole day traffic, while the second one is a zoom to some of the main intervals detected. Hotmail is shown since it is the service with more disruption events detected by the algorithms. The intervals of unavailability detected by the algorithm are drawn also. The first plot shows a full day of traffic and the second one zooms to 1 hour around the previous discussed event.

It can be seen that the amount of traffic suggests the service is working and the gap around 15:35:10 is clearly visible. After these period without traffic the service seems to reestablish normally, creating a traffic peak after the detected problem that reaches almost 5 MBps.

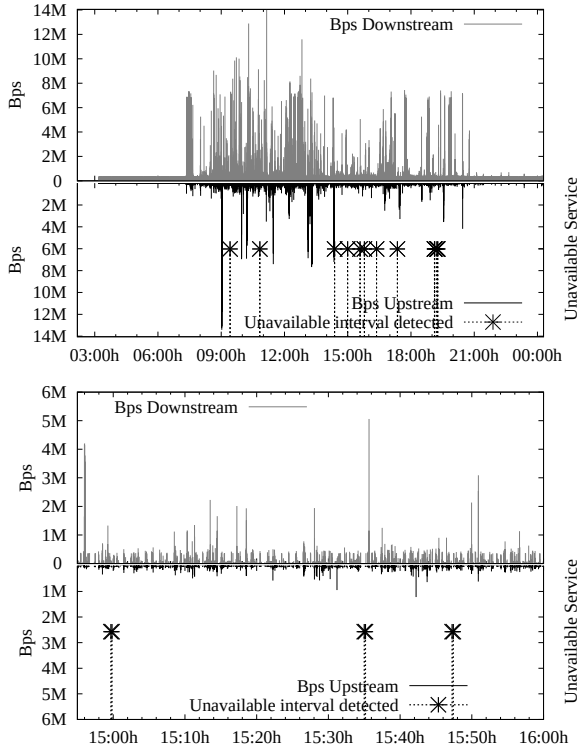


Figure 9. Bps for the use of Hotmail service by the university community

VII. Conclusions

In this paper, a simple algorithm to detect periods of unavailability services has been presented. It is based only on passive capture of traffic.

Although there are more service monitoring software available, to the authors best knowledge, they are based on active probing systems. Active monitoring presents some disadvantages, which may discourage network administrators of its use in scenarios where the impact of the monitoring needs to be minimized. First, because it requires to check if a service is available it would imply to make periodically requests to different servers. In some scenarios, like high loaded servers or monitoring third party services it is not possible to make these requests as frequently as needed, in order to avoid overhead or security alarms. Apart from that, the probing requests should be chosen carefully in order to avoid problems with proxy caches, which could give the impression that the service is working properly while other clients would not be able to use the service. Another problem is the difficulty to select a location for monitoring clients in multiple subnet scenarios. In these cases, at least a pair of clients should be placed in each subnet in order to detect possible problems inside. Moreover, every client should be clock synchronized in order to report coherent times with the rest of monitoring clients.

As the proposed model is passive and based only on the study of packet counts between servers and clients it will not interfere with the traffic on the network. Therefore, any problem of interference, monitoring client overload, network problems with measured server availability is avoided.

Although the algorithm is based on the expected behaviour of the transfers between client and servers seen at packet level, it has been shown how different kinds of disruptions, as for instance, an unexpected server reboot has an influence on the network traffic at packet level captured at client's side.

Another advantage of using the proposed model is that it is based in a single location. That means the measure is not dependent on the location of multiple monitoring agents. The network administrator has just to select an appropriate passive observing location, where it can see the traffic between the population of clients to monitor and the servers of interest. This is a much simpler decision that can be typically solved by placing the sniffer at organization's network's edge.

VIII. Future work

Currently, we are working to extend the algorithm to detect service failures without focusing on specific servers, just by analyzing sniffed traffic and applying the current algorithm to every connection seen. In this manner, the algorithm can work as an service anomaly detection system that warns administrator of service issues. This is useful in large organizations that may not have a clear list of services accessed by users but, nevertheless, need to react to service unavailability problems.

An improvement that can be implemented in order to reduce the number of false positives, is to use the two membership functions described in the algorithm to apply some method of fuzzy logic.

ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Science and Innovation through the research project INSTINCT (TEC-2010-21178-C02-01). Also, the authors want to thank Public University of Navarra for funding through PIF grant.

REFERENCES

[1] I. Prieto, M. Izal, E. Magana, and D. Morato, "Detecting disruption periods on tcp servers with passive packet traffic analysis," in SOFTENG 2015, The First International Conference on Advances and Trends in Software Engineering, April 2015, pp. 34–40.

[2] "NAGIOS, a commercial-grade network flow data analysis solution," 2009-2015. [Online]. Available: http://www.nagios.com/ [accessed: 2015-01-30]

[3] "ZABBIX, the ultimate enterprise-level software designed for monitoring availability and performance of it infrastructure components," 2001-2014. [Online]. Available: http://www.zabbix.com [accessed: 2015-02-02]

[4] "CACTI, a complete network graphing solution." 2004-2012. [Online]. Available: http://www.cacti.net/ [accessed: 2014-12-29]

[5] "MUNIN, networked resource monitoring tool," 2003-2013. [Online]. Available: http://munin-monitoring.org/ [accessed: 2015-01-15]

[6] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tiered web applications using queueing predictor," in Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, 2006, pp. 106–114.

[7] D.-J. Lan, P. N. Liu, J. Hou, M. Ye, and L. Liu, "Service-enabled automatic framework for testing and tuning multi-tier system," in e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on, 2008, pp. 79–86.

[8] D. Chua, E. Kolaczyk, and M. Crovella, "Efficient monitoring of end-to-end network properties," in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 3, 2005, pp. 1701–1711.

[9] Y. Park, "Systems monitoring using petri nets," in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, vol. 4, 1997, pp. 3245–3248.

[10] C. H. Choi, M. G. Choi, and S. D. Kim, "CSMonitor: a visual client/server monitor for corba-based distributed applications," in Software Engineering Conference, 1998. Proceedings. 1998 Asia Pacific, 1998, pp. 338–345.

[11] C. Steigner, J. Wilke, and I. Wulff, "Integrated performance monitoring of client/server software," in Universal Multiservice Networks, 2000. ECUMN 2000. 1st European Conference on, 2000, pp. 395–402.

[12] G. Song, "The study and design of network traffic monitoring based on socket," in Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on, 2012, pp. 845–848.

[13] G. Fang, Z. Deng, and H. Ma, "Network traffic monitoring based on mining frequent patterns," in Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on, vol. 7, 2009, pp. 571–575.

[14] A. Tachibana, S. Ano, and M. Tsuru, "Selecting measurement paths for efficient network monitoring and diagnosis under operational constraints," in Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on, 2011, pp. 621–626.

[15] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 1, 2003, pp. 134–144.

[16] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in International Networking Conference (INET) '98. Geneva, Switzerland: The Internet Society, Jul 1998, pp. 1–1.

[17] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," in Proceedings of the 2011 31st International Conference on Distributed Computing Systems, ser. ICDCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 310–321. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2011.27

[18] D. Schatzmann, S. Leinen, J. Kgel, and W. Mhlbauer, "FACT: Flow-based approach for connectivity tracking," in Passive and Active Measurement, ser. Lecture Notes in Computer Science, N. Spring and G. Riley, Eds. Springer Berlin Heidelberg, 2011, vol. 6579, pp. 214–223.

[19] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," vol. PP, no. 99, 2014, pp. 1–1.

[20] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Understanding passive and active service discovery," in Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 57–70. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298314

[21] G. Klir and B. Yuan, Fuzzy sets and fuzzy logic. Prentice Hall New Jersey, 1995, vol. 4.

Deriving Robust Distributed Business Processes with Automated Transformations of Fallible Component Processes

Lei Wang,
Luís Ferreira Pires
and Marten J. van Sinderen

CTIT, University of Twente,
the Netherlands
Emails: {l.wang-1, l.ferreirapires, m.j.vansinderen}@utwente.nl

Andreas Wombacher

Achmea, the Netherlands
Postbus 866
3700 AW Zeist
Email: andreas.wombacher@achmea.nl

Chi-Hung Chi

CSIRO, Australia
3-4 Castray Esplanade,
Hobart, Tasmania, 7000
Email: chihungchi@gmail.com

***Abstract*—Due to the possibility of system crashes and network failures, the design of robust interactions for collaborative business processes is a challenge. If a process changes state, it sends messages to other relevant processes to inform them about this change. However, server crashes and network failures may result in a loss of messages. In this case, the state change is performed by only one process, resulting in global state/behavior inconsistencies and possibly deadlocks. Our idea to solve this problem is to (automatically) transform the original processes into their robust counterparts. The robust initiator process re-tries failed interactions by resending the request message. The robust responder then replies with the possibly lost response message, without processing the request again. We illustrate our solution using a subset of Web Services Business Process Execution Language (WS-BPEL). A WS-BPEL process is modeled using a so called Nested Word Automata (NWA), to which we apply our transformation solution and on which we perform correctness proof. We have also analyzed the performance of our prototype implementation. In our previous work, we assumed that a certain pre-defined interaction follows the failed interaction. In this work, we lift this limitation by allowing an arbitrary behavior to follow the failed interaction, making our solution more generally applicable. This paper is an extension of our previous paper [1]. The additional contents of this paper is the following.**

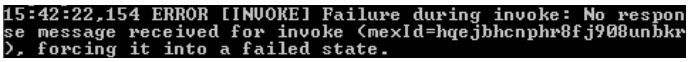
Keywords–robust; collaborative processes; control flow; WS-BPEL; interactions; system crash; network failure; automata.

I. INTRODUCTION

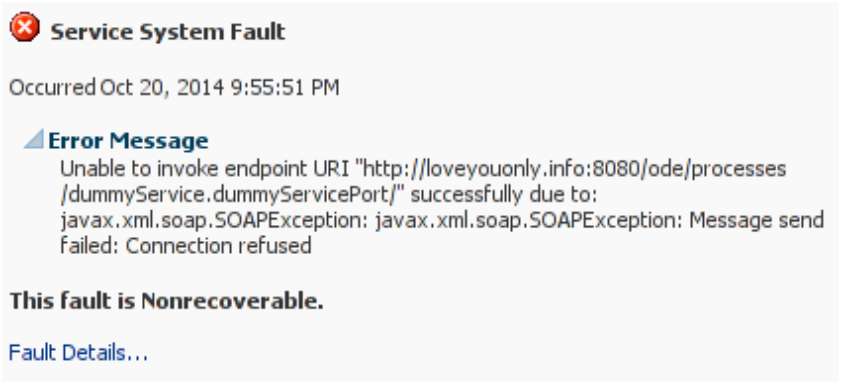
The electronic collaboration of business organizations has grown significantly in the last decade. Often data interchange is based on processes run by different parties exchanging messages to synchronize their states. If a process changes state, it sends messages to other relevant processes to inform them about this change. However, server crashes and network failures may result in a loss of messages. In this case, the state change is performed by one process, resulting in global state/behavior inconsistencies and possible deadlocks. In general, a state inconsistency is not recovered by the process engine that executes the process. This can be seen from a screen dump of errors after a system crash of the process engines such as Apache ODE and Oracle BPM, as is shown in Figure 1. Figure 1a shows that in the Apache ODE process engine the initiator sends the message to an unavailable server. Figure 1b shows that in the Apache ODE process engine the initiator sends a request message, and the responder crashes without sending the response message. Figure 1c shows the Oracle process engine 12c, which sends message to an unavailable server (see Figure 1).

Figure 2a shows that normally, a business process is deployed to a process engine, which runs on the infrastructure services (OS, database, networks, etc.), where system crashes and network failures may happen. Our solution to recover from failures is to transform business processes into their robust counterparts, as shown in Figure 2b. The robust process is deployed on the unmodified infrastructure services and is recoverable from some interaction failures caused by system crashes and network failures. Our solution has the following properties: (1) the application protocols are not modified. We do not modify the message format nor message sequence, e.g., by adding message fields that are irrelevant for the application logic or adding acknowledge messages to the original message sequence. The service autonomy is kept in that if one party transforms the process according to our approach and the other party does not, they can still interact with each other, although without being able to recover from system crashes and network failures. (2) the process transformation is transparent for process designers. (3) the solution conforms to the process language specification. We use the standard process language constructs without extending the language to avoid making the robust process depend on a specific engine. In this paper, we illustrate our solution using WS-BPEL. WS-BPEL is a language for specifying business process behavior

(a) Service unavailable

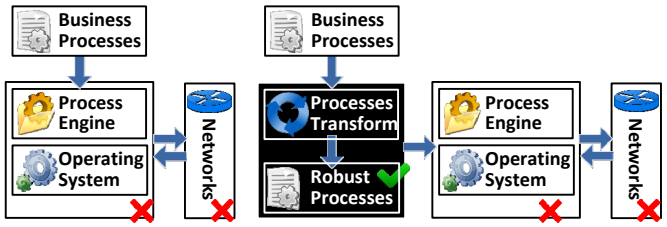


(b) Pending response



(c) Service unavailable

Figure 1. Interaction failures



(a) Interaction failures (b) Robust process transformation

Figure 2. Our idea to cope with failures.

based on web services. As an OASIS standard, it is widely used by enterprises. However, other process languages may be applicable as long as they support similar workflow patterns.

This paper is an extension of our previous paper [1]. The additional contents of this paper is the following.

1) We analyze the possible synchronization failures in more detail.
2) More Nested Word Automata models of WS-BPEL activities are described inthis paper,suchas, “invoke”, “receive” and “reply” activities.
3) Omitted information on the correctness criteria and correctness evaluation process is presented in this paper.
4) Process transformation complexity of our soultion is analyzed.

This paper is based on our previous work [2][3][4][5], where we assumed that a certain pre-defined interaction follows the failed interaction, i.e., the only sequence control is assumed that the further interaction is sequentially following the failed interaction. In this paper, we lift this limitation by allowing an arbitrary behavior to follow the failed interaction, making our solution more generally applicable. We support conditional control flow and loops and their arbitrary combination as possible further interaction after interaction failure. The structure of the paper is the following: Section II analyzes possible interaction failures. Section III proposed our process transformation-based solution. Section IV validates our solution. Section V discusses related work and Section VI concludes our paper.

II. Interaction Failure Analysis

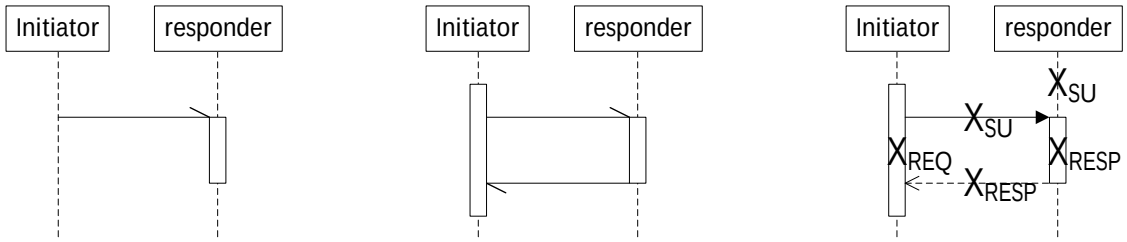
This section analyzes possible interaction failures of collaborative processes caused by system crashes and network failures.

A. Process Interaction Patterns

Process interaction failures are specific to interaction patterns. In [6], 13 interaction patterns are identified. In this paper, we focus on the *send*, *receive* and *send-receive* patterns. This limitation is not severe because more complex patterns can be composed using these basic interaction patterns. Figure 3a shows an initiator that sends a one-way message to a responder. The initiator behavior corresponds to the *send* pattern, while the responder behavior corresponds to the *receive* pattern. In pattern *send-receive* in Figure 3b the initiator combines one *send* and one *receive* pattern. We call this pattern asynchronous interaction in the sequel of the paper. In Figure 3c, the initiator starts a synchronous interaction by sending a request and getting a response, which characterize the *send-receive* pattern.

B. Process Interaction Failures

Table I shows a failure classification scheme [7]. Crash failure, omission failure and timing failure are considered in this work. Crash failure is referred as *system crashes* in this paper. Omission failure and timing failure occur when the network fails to deliver the messages (in a specified time interval) and are referred to as *network failures* in this paper. However, response failures due to a flaw in the process design and arbitrary failure, also referred to as Byzantine failure, which is more of a security issue, are out of the scope of this work.



(a) *send* and *receive* (b) *send-receive* case I (c) *send-receive* case II

Figure 3. Process interaction patterns.

TABLE I. Failure Scheme.

Type of failure	Description
Crash failure	Server halts, bus is working correctly until it halts.
Omission failure	Server fails to respond to incoming requests.
Receive omission	A server fails to receive incoming messages.
Send omission	A server fails to send messages.
Timing failure	Server's response lies outside the specified time interval.
Response failure	Server's response is incorrect.
Value failure	The value of the response is wrong.
State transition failure	The server deviates from the correct flow of control.
Arbitrary failure	Server may produce arbitrary responses at arbitrary times.

Interaction failures caused by system crashes and network failures are *pending request failure*, *pending response failure* and *service unavailable* [3]. As all failures possible in the interaction patterns of Figure 3a and Figure 3b are covered by Figure 3c, we look only into the interaction failures of the interaction pattern in Figure 3c. *Service unavailable* (marked as X_{SU}) is caused by a responder system crash or a network failure of the request message delivery. At process level, the initiator is aware of the failure through a catchable exception of the process implementation language. *Pending request failure* (marked as X_{REQ}) is caused by initiator system crashes after sending a request message. The initiator is informed of the failure after restart, e.g., through catchable exceptions. However, the responder is not aware of the failure, so that it replies with the response message and continues execution. *Pending response failure* (marked as X_{RESP}) is caused by a responder system crash or a network failure of the response message delivery. In both cases, the responder replies with the response message (after a restart if the responder system crashes) and continues execution. However, the connection gets lost and the initiator cannot receive the response message. The initiator is aware of this failure after a timeout.

C. Assumptions concerning the failure behavior

Due to the heterogeneous infrastructure, e.g., different process engine implementations or network environments, we have to make the following assumptions concerning the failure behavior of the infrastructure:

1) Persistent execution state. The state of a business process (e.g., values of process variables) are kept persistent and survive system crashes.

2) Atomic activity execution (e.g., invoke, receive, reply). A system crash means that the execution is stopped only after the previous activity is finished and the next activity has not started. A restart means that execution resumes from the previous stopped activity. These assumptions correspond with the default behavior of the most popular process engines, such as Apache ODE or Oracle BPEL Process Manager (released as a component of Oracle SOA Suite). In Apache ODE's term, this is named as persistent processes in their default configuration. Otherwise, this configuration can be modified to "in-memory" at deployment time [8]. For Oracle BPEL Process Manager, this is named as "durable" processes, otherwise is named as "transient" processes. By default all the WS-BPEL processes are durable processes and their instances are stored in the so called dehydration tables, which survives system crashes [9].

3) Network Failures interrupt the established network connections and the messages that are in transit get lost.

III. Process Transformation Based Solution

Figure 4 shows our approach based on process transformation. The transformation is implemented at an abstract level by using NWA (Nested Word Automata) [10]. NWA is an extension of the classical automata, which maintains the nested syntax of WS-BPEL process. Furthermore, its automata based definition, which facilitates the description of reliable interaction principles, correctness proof and complexity analysis formalized. Figure 4 shows that the process transformation is divided into three steps: *Transform1* transforms a business process into a NWA, *Transform2* transforms the NWA to the NWA' of a robust process, and finally, *transform3* generates the robust process from the NWA'.

A. Business Processes

We choose WS-BPEL [11] as process specification language in our work. However, other process languages may be applicable as long as they support similar workflow patterns [12]. A WS-BPEL process is a container where relationships to external partners, process data and handlers for various purposes and, most importantly, the activities to be executed are declared. As an OASIS standard, it is widely used by enterprises. We use Nested Word Automata (NWA) [10] to

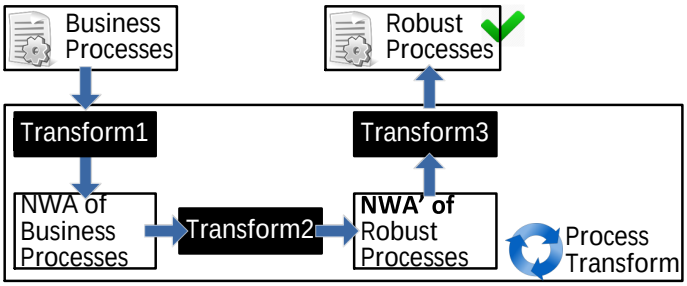


Figure 4. Process transformation

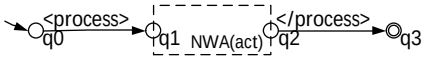


Figure 5. NWA model of a process.

describe the underlying semantics of WS-BPEL and use them as a basis for our formal evaluation. We choose NWA because we need to model the nested structure of WS-BPEL syntax. While traditional finite state automata can be used for describing all possible states of messages, and their sending and receiving sequences, they lack the capability of describing nested structures of activities.

An NWA is an automaton that has hierarchical nesting structures. Formally, an NWA A over an alphabet Σ is a structure $(Q, q_0, Q_f, P, p_0, P_f, \delta_c, \delta_i, \delta_r)$ consisting of

- a finite set of (linear) states Q,
- an initial (linear) state $q_0 \in Q$,
- a set of (linear) final states $Q_f \subseteq Q$,
- a finite set of hierarchical states P,
- an initial hierarchical state $p_0 \in P$,
- a set of hierarchical final states $P_f \subseteq P$,
- a call-transition function $\delta_c : Q \times \Sigma \mapsto Q \times P$,
- an internal-transition function $\delta_i : Q \times \Sigma \mapsto Q$, and
- a return-transition function δ_r: $Q \times P \times \Sigma \mapsto Q$.

The definition of Q, q_0, Q_f, δ_i corresponds to the definition of a finite state automata over an alphabet Σ [13]. The alphabet Σ represents all possible process behaviors, e.g., $<process> \in \Sigma$ represents the starting of a business process, $?m_i \in \Sigma$ represents receiving a message while $!m_j \in \Sigma$ represents sending a message. An internal transition $\delta_i(q_i, !m_i) = q_j$ represents that the process replies a message $!m_i$ at the state q_i and then enters the state q_j. The hierarchical states P, p_0, P_f are used to describe the nesting structure of an NWA. A *call transition* δ_c enters the nested automaton while a *return transition* δ_r leaves the nested automaton. The other transitions are also found in traditional automata, and called internal transitions. A nested structure is graphically represented as dashed box. The NWA model of a WS-BPEL process is shown in Figure 5. A call transition $\delta_c(q_0, <process>) = (q_1, p_a)$ starts from the initial state and a return transition $\delta_r(q_2, p_a, </process>) = q_3$ leads to the accepted state. The NWA model of an activity *NWA(act)* is nested within the NWA of the process. This is described by the hierarchical state p_a.

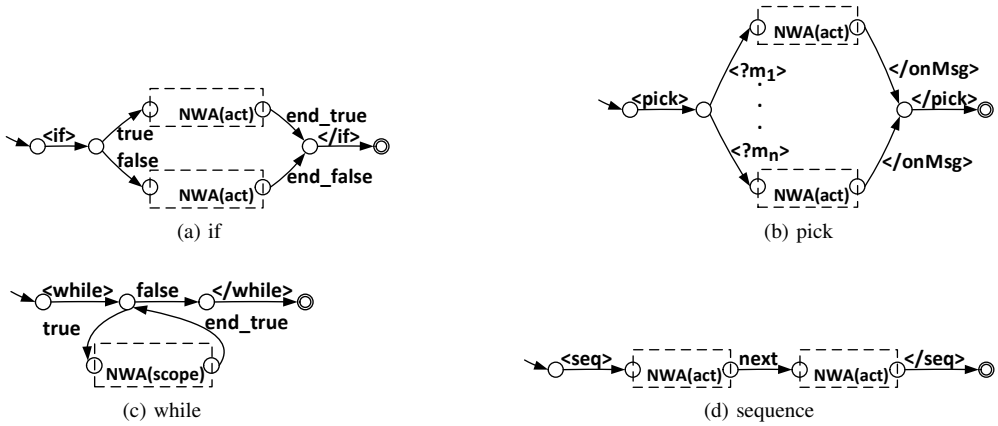
WS-BPEL activities are divided into two categories, namely *basic* and *structured* activities. The currently supported *structured* activities are *if*, *pick*, *while* and *sequence*, as shown in Figure 6. The conditional branch (*if*), repeat (*while*), and sequential (*sequence*) activities are modeled as shown in Figure 6a, 6c, 6d, respectively. Figure 6b shows the model of a *pick*, which is another case of conditional control flow that depends on the type of the incoming message. The *flow* (concurrent execution) or the other forms of loops (*RepeatUntil, ForEach*) are not considered now and will be considered in future work. Each structured activity model has exactly one call transition and one return transition to *enter* and *leave* its nested structure(s), which is represented as a dash box.

The models of *basic* activities are visualized in Figure 7. These models can be *nested* (to replace the dash box in Figure 6) in the models of structured activities. However, the model of *exit* is an exception. Once the transition *exit* fires, the NWA goes to a terminated state. From this state, the NWA is *deactivated* (no transition will leave this state), which breaks the well-nested structure.

B. Transformation Method

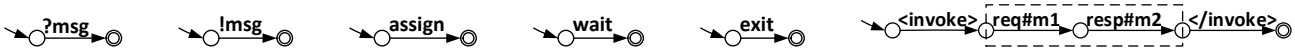
An operation that can be safely repeated is called idempotent [7]. Idempotent operations can be recovered by resending the request message. However, a request resent to non-idempotent operations (such as bank transfer operations) triggers potentially incorrect executions. Our solution for non-idempotent operation is that when a failure happens, a resent message is replied with a copy of the previous processing result.

In the example of Figure 8a, the WS-BPEL snippet receives a message $m1$, performs some (non-idempotent) processing, then replies with a message $m2$. The next incoming messages could be $m3$ or $m4$. If the initiator sends request $m3$ or $m4$, this implies that the initiator has successfully received the response message $m2$. If due to an interaction failure, for example, the initiator crashes and fails to receive the response message $m2$, the initiator can recover by resending request message $m1$. Thus, the responder can be aware of whether failures have happened by inspecting the incoming message. If the incoming message is a resent message, this implies that a failure happened in the previous interaction. In this section, we will first present our formal method of calculating the set of failure-resent messages at a state. We then transform the responder process to use a copy of the previous result as response. Figure 8b shows that in order to make a copy of the response message, we use an *assign* activity to keep the result value in a process variable $\$copy$. In the *pick* activity, we add an *onMessage* branch to accept the resent message $m1$ and use the variable $\$copy$ as the response. However, a resent message could be sent multiple times before the response is ultimately received. We nest the *pick* activity in a *while* to cope



(a) if (b) pick (c) while (d) sequence

Figure 6. NWA model of WS-BPEL structured activities.



(a) receive (b) reply (c) assign (d) wait (e) exit (f) invoke

Figure 7. NWA model of WS-BPEL basic activities.

```
<sequence>
   <receive variable="m1" ... />
   <!-- some process -->
   <reply variable="m2" ... />
   <pick>
      <onMessage variable="m3".../>
      <onMessage variable="m4".../>
   </pick>
</sequence>
```

(a) example WS-BPEL snippet

```
<sequence>
   <receive variable="m1" ... />
   <!-- some process -->
   <reply variable="m2" ... />
   <assign>$copy := $m2</assign>
   <pick>
      <onMessage variable="m3".../>
      <onMessage variable="m4".../>
      <onMessage variable="m1"…>
         <reply variable="copy" … />
      </onMessage>
   </pick>
</sequence>
```

(b) transformation, step I

Figure 8. WS-BPEL example of our solution.

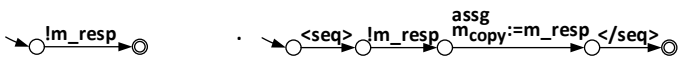
with the duplicate resent message. Our process transformation algorithm is presented as follows.

1) Responder Transformation Algorithm: For a WS-BPEL process, given its *NWA* model $(Q, q_0, Q_f, P, p_0.P_f, \delta_c, \delta_i, \delta_r)$ over the alphabet Σ, we assume that the alphabet that represents the response messages is Σ_{resp} and the alphabet that represents the request messages is Σ_{req}, thus $\Sigma_{req} \subseteq \Sigma$ and $\Sigma_{resp} \subseteq \Sigma$. The transformation algorithm is as Figure 9.

The algorithm iterates through all combinations of a state q, a request message $?m_{req}$ and a response message $!m_{resp}$. In line 2, we check if the message pair $(?m_{req}, !m_{resp})$ corresponds to the request and response for a synchronous operation and at state q, the response message $!m_{resp}$ is sent, represented by a transition $\delta_i(q, !m_{resp})$. This is the failure point that the response message may be lost due to interaction failures and where our transformation method applies. As defined in line 3, we first make a copy of the response message, as shown in Figure 10. The NWA model of *reply* activity in Figure 10a is replaced by an NWA model of a *sequence* activity in Figure 10b, in which a *reply* activity model and an *assign* activity model are nested. The *assign* activity model represents

```
1:  for all q ∈ Q, ?m_req ∈ Σ_req and !m_resp ∈ Σ_resp do
2:     if (m_req, m_resp) is a synchronous message pair and
       δ_i(q, !m_resp) is defined in NWA then
3:        save_reply(q, !m_resp)
4:        N ← next_receive(q, !m_resp)
5:        for all ?m_next ∈ N and q_next ∈ Q do
6:           if δ_i(q_next, ?m_next) is defined in NWA then
7:              transform_receive(q_next, ?m_next)
8:           else if δ_c(q_next, ?m_next) is defined in NWA then
9:              transform_pick(q_next, ?m_next)
10:          end if
11:       end for
12:    end if
13: end for
```

Figure 9. Responder process transformation algorithm



(a) *reply* activity (b) Transformed *reply activity*

Figure 10. Responder process transformation, *reply* activity.

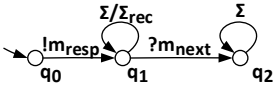


Figure 11. The automaton $A(!m_i, ?m_{next})$.

the copy of the reply message into the variable m_{copy}. In order to process the possible resent request message $?m_{req}$ due to the lost of the message $!m_{resp}$ sent at state q, we calculate the set of all possible next incoming messages, which is defined as $next\_receive(q, !m_{resp})$ in line 4. We construct an automaton $A(!m_{resp}, ?m_{next})$ as in Figure 11 to describe that a process replies with a message $!m_{resp}$ and waits for some possible next incoming message $?m_{next}$. $\delta(q_0, !m_{resp}) = q_1$ models the reply of the response message $!m_{resp}$. $\delta(q_1, \Sigma/\Sigma_{req}) = q_1$ represents some process execution in which no messages are received. $\delta(q_1, ?m_{next}) = q_2$ represents that the process receives an incoming message $?m_{next}$. $\delta(q_2, \Sigma) = q_2$ models any process execution. For the process *NWA* model, at some state q, a reply of a message $!m_{resp}$ is represented by an internal transition $\delta_i(q, m_{resp})$. We change the initial state of the process *NWA* model to from q_0 to q, and call this automaton *NWA*(q). Starting at q, after replying the message $!m_{resp}$, if one possible next incoming message is $?m_{next}$, then *NWA*$(q) \cap A(!m_{resp}, ?m_{next}) \neq \emptyset$, i.e., the process modeled by *NWA* has the behavior described by $A(!m_{resp}, ?m_{next})$.

The intersection operation $\cap$ between an *NWA* and an finite state automaton is defined to check whether the business process modeled by the *NWA* has the message sending and receiving behavior modeled by the automaton. The intersection operation is based on finite state automata. We "flatten" an *NWA* to a finite state automaton by skipping hierarchical information, described as follows. Given a NWA $(Q, q_0, Q_f, P, p_0, P_f, \delta_c, \delta_i, \delta_r)$ over the alphabet Σ, the "flattened" automaton is $A(Q, q_0, Q_f, \Sigma, \delta)$, where Q, q_0, Q_f and Σ are the same as the NWA, the transition function δ is defined as

1) $\delta(q_{i1}, a) = q_{i2}$, if the NWA has an internal transition $\delta_i(q_{i1}, a) = q_{i2}$.
2) $\delta(q_{c1}, a) = q_{c2}$, if the NWA has a call transition $\delta_c(q_{c1}, a) = (p, q_{c2})$.
3) $\delta(q_{r1}, a) = q_{r2}$, if the NWA has a return transition $\delta_r(q_{r1}, p, a) = q_{r2}$.

Both call transitions and return transition are treated as flat transitions that the hierarchical state p is not considered. The intersection operation can be done between two finite state automata, as defined in [13].

We define the set of all possible next incoming messages as $next\_receive(q, !m_{resp}) = \{?m_{next} | ?m_{next} \in \Sigma_{req} \wedge$ *NWA*$(q) \cap A(!m_{resp}, ?m_{next}) \neq \emptyset\}$.

For all $?m_{next} \in next\_receive(q, !m_{resp})$ and $q_{next} \in Q$, if at the state q_{next} the next incoming message $?m_{next}$ is received, two cases of transition may be defined in *NWA*: in a model of a *receive* activity as an internal transition $\delta_i(q_{next}, ?m_{next})$ or in the model of a *pick* activity as a call transition $\delta_c(q_{next}, ?m_{next})$. For the first case (line 6), as shown in Figure 12a, the procedure $transform\_receive(q_{next}, ?m_{next})$ is introduced as follows. We replace the transition with a *pick* activity with two branches, as shown in Figure 12b. One *onMessage* branch models the receive of the resent message $?m_{req}$ and the reply of the result message m_{copy}. The other *onMessage* branch models the receive of the message $?m_{next}$, and after that we set the flag $success$ to $true$ to indicate that the previous interaction is finished successfully.

However, a possible loss of the response message m_{copy} triggers multiple resending of the request m_{req}. Therefore, the *pick* activity is defined in a *while* iteration so that multiple requests $?m_{req}$ can be accepted. Figure 12c shows that the *while* iteration ends when the flag $success$ is set to *true*.

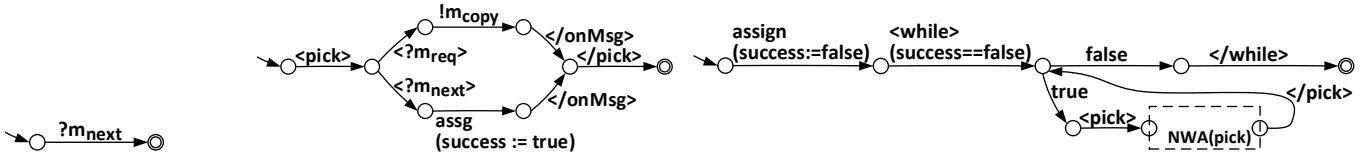
For the second case (line 8), as shown in Figure 13a, the message $?m_{next}$ is one of the messages in $m_1, ..., m_n$. Figure 13b shows that we then add a call transition $<?m_{req} >$ to model that the process accepts the resent message, and an internal transition $!m_{copy}$ to represent the reply using a copy of the previously cached result m_{copy}. In the other branches, the nested NWA(act) is replaced by the model of a *sequence* activity, in which we model the assignment of the flag variable *success* to *true*, followed by the original NWA(act). Similarly, in order to cope with a possible loss of the response message m_{copy}, the *pick* activity model is nested in a *while* iteration to handle multiple resent messages, as shown in Figure 12c. We do not directly prove the correntness of this algorithm, however, the correctness of the transformed processes by this algorithm have been proved in Section IV.

After the transformation, at some states the responder can receive more messages than the original process, because the resent message can be accepted and be replied. However, the request is not processed again. In this sense, we do not give malicious initiators any chance of jeopardizing the process by changing the sequence of requests or sending the same request multiple times.

C. Initiator Transformation

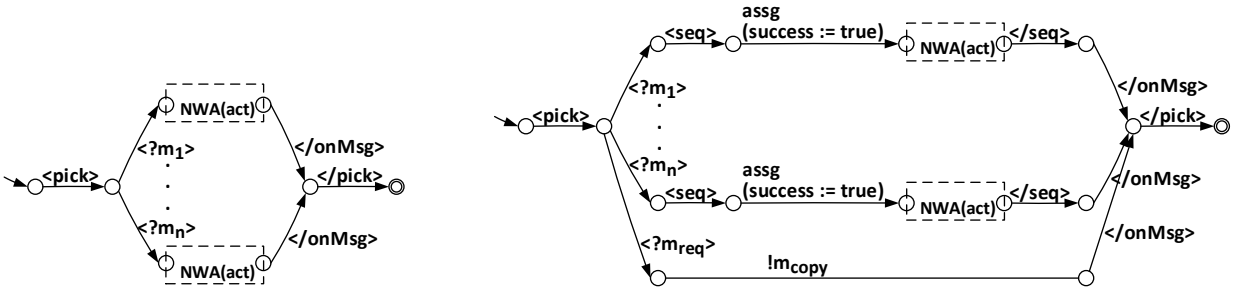
The initiator starts the interaction by executing the *invoke* activity. An *invoke* activity, which is shown as Figure 14a, is replaced by the model of a scope activity, which consists of an NWA of a fault handler and a *sequence* activity model. Nested in the *sequence* activity model there is the model of the original *invoke* activity, followed by an assignment of the $false$ value to a process variable w. The whole NWA of the *scope* activity is nested in a *while* activity model.

The whole model represents the process behavior of invocation. If failure happens and gets caught by a fault handler, the process waits for a specific time period and finishes the *scope* activity, then the outside *while* makes the *invoke* activity be executed again, until it finishes successfully and the variable w is assigned to $false$. The possible interaction failure is modeled as the transition *fail*. This is a reasonable failure model since that if a failure happens the control flow is deviated from the normal flow to the end of the scope to which a fault handler is attached, rather than leading the process to an exceptional end.



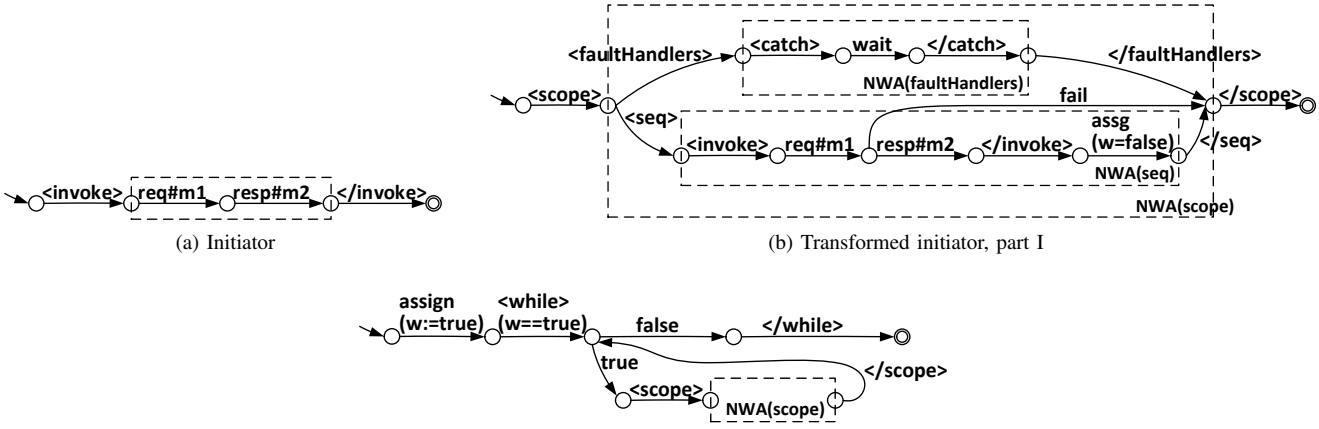
(a) Responder, *receive* activity (b) Transformed responder, part I (c) Transformed responder, part II

Figure 12. Responder process transformation, *receive* activity.



(a) Responder, *pick* activity (b) Transformed responder, part I

Figure 13. Responder process transformation, *pick* activity.



(a) Initiator (b) Transformed initiator, part I

(c) Transformed initiator, part II

Figure 14. Initiator process transformation.

1) Recoverable Assumption: Assume that $(?m_{req}, !m_{resp})$ is a pair of synchronous request and response messages, the process receives request message $?m_{req}$, then at state q, the process sends the response message $!m_{resp}$. However, if $?m_{req} \in next\_receive(q, m_{resp})$, then one of the next possible messages is still $?m_{req}$, in this case, the responder cannot distinguish a resent message due to a failure from a normal request message. Thus, we have to require that in the process design the condition $?m_{req} \notin next\_receive(q, !m_{resp})$ can be met. However, by following a few process design principles during the design of the original process, this condition can be met. An example is a split of message $?m_{req}$ into two different messages, $?m_{req1}$ and $?m_{req2}$ (for example, one message is used to send request, the other asks for results). The initiator sends the two messages back to back. If a responder receives $?m_{req1}$, then it waits for $?m_{req2}$, rather than waiting for $?m_{req1}$ again.

IV. Evaluation

This section presents the correctness validation of our solution. We also evaluate the performance overhead of our prototype under different workloads. Finally, we analyze the complexity of the transformed process by comparing it with the original process.

A. Correctness Validation

The correctness proof shows that the solution introduced in Section III provides a robust process. The core of the proof is to define the correctness criteria for asynchronous and synchronous interactions and represent them such that they can be automatically evaluated.

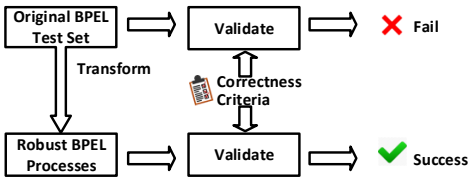


Figure 15. The setup of the correctness proof.

Figure 15 shows the setup of our correctness proof. We take the test set of 726 example WS-BPEL processes, which implement all possible Internet Open Trading Protocol (IOTP) interactions [14], and we transform them into the *NWA* model of the corresponding robust business process. The proof is finished by checking that the *NWA* process model is a subset of the correctness criteria, which are modeled as automata. Given an automaton $A(Q, \Sigma, \delta, q_0, F)$, each state in Q represents a state of the messages sending and receiving status. Set Σ models of all possible process behaviors, e.g., sending and receiving messages. δ is transition function: $Q \times \Sigma \rightarrow Q$ that models the state transition triggered by process execution, e.g., for states $q_i, q_j \in Q$ and $!m \in \Sigma$, $\delta(q_i, !m) \rightarrow q_j$ represents that at state q_i, the process replies with message $!m$ and then enters state q_j. The correctness criteria automata models the set of correct message sending and receiving sequences. We present the correctness criteria as follows.

1) Initiator Side Correctness Criteria: There are two criteria due to the interaction patterns: the criteria for a single message sending and the criteria for synchronous request and response message pair. In this paper, we discuss only the latter due to the page limitations. The automaton is visualized as Figure 16a. A correct interaction is regarded as, a request message $?m1$ can be sent at state q_0 and can be resent multiple times at state q_1 until a response message $!m2$ is received at state q_2. The correctness criteria automaton is formally defined as $\{Q, q_0, Q_f, \Sigma, \delta\}$, where

- $\Sigma = \{?m1, !m2\}$. $?m1$ and $!m2$ are the synchronous request and response messages.
- $Q = \{q_0, q_1, q_2\}$ and $Q_f = \{q_2\}$.
- For the transition $\delta(q_0, ?m1) = q_1$, the state changes from q_0 to q_1 on the input $?m1$ and stays on state q_1 on the input $?m1$ (transition $\delta(q_1, ?m1) = q_1$). This represents that a request could be sent multiple times due to failures. Transition $\delta(q_1, !m2) = q_2$ represents that if the response is received the initiator is in an accepted state and any transition is accepted (transition $\delta(q_2, \Sigma) = q_2$).

2) Responder Side Correctness Criteria: The property we want to validate is that any resent message can be accepted by the transformed process and replied. Given a process, assume that the synchronous request and response messages are m_j and m_i. If the transformed robust process model is NWA', for state q_i and transition $!m_i$, we have $?m_j \in next\_receive(q_i, !m_i)$. The idea behind it is that after the reply message $!m_i$ is sent, the robust process can accept possible resent messages due to failures. In this case, the response should be sent without reprocessing. The criteria are shown as Figure 16b.

The process control flows can be designed in arbitrary ways, and since we cannot exhaust all possibilities, we use a WS-BPEL test set that implements all possible IOTP interactions, which is a total of 726 BPEL processes. After the transformation of the test processes into automata, we apply the subset check to evaluate the correctness of the WS-BPEL test processes, i.e., we prove that for all processes and their *NWA* model and criteria automaton A, *NWA* $\in A$, i.e., all messages sending and receiving sequences are correct.

We take the process in Figure 8 to illustrate the correctness validation. An original WS-BPEL snippet shown in Figure 8a is transformed into the robust counterpart, and their automata models are shown as Figures 17a and 17b, respectively. At state q_2, where the message $!m_2$ is to be replied, the set of the possible next incoming messages of the responder is $next\_receive(q_2, !m2) = \{?m1, ?m3, ?m4\}$. The criteria for the synchronous request $?m1$ and the response $!m2$ is shown as Figure 18. First, we do a subset check to prove that the original is not a subset of the criteria. Actually, we can see that the message sequence $(.., ?m1, !m2, .., ?m1, !m2, ..?m3, ..)$ can be accepted by the criteria automaton. However, this sequence cannot be accepted by the model of the original process, since there is no transition defined for the second $?m1$. Second, we do a subset check to prove the transformed automata model is a subset of the criteria, i.e., all sending and receiving message sequences are correct.

B. Performance Evaluation

In Figure 2 of the whole setup, if the infrastructure (OS, process engine, hardware and network configuration) is the same, performance depends mainly on the process design and the workload, i.e., $performance = Test(ProcessDesign, workload)$.

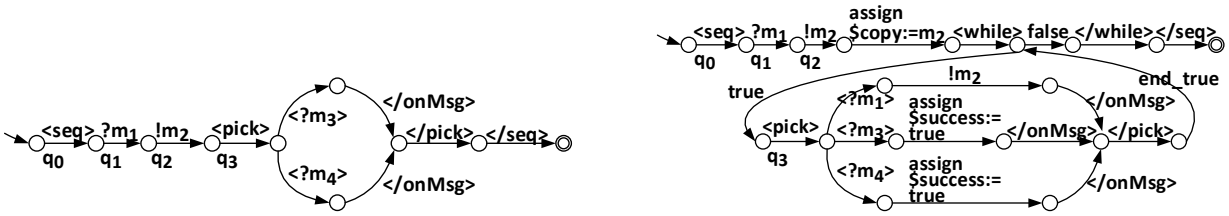
We use similar setup of our performance test with our previous performance tests [15], which is shown in Figure 19. We use the cloud infrastructure from Amazon EC2. The initiator and responder processes are deployed on two computing instances and we use a local client to collect the performance data. At the node *initiator*, the original and transformed initator processes are deployed. At the node *responder*, the original and transformed responder processes are deployed. At the node *performance data collector*, a local client is deployed.

We evaluated the performance overhead of our transformed process under different workloads. The number of requests sent per minute by the local client complies with a Poisson distribution with parameters $\lambda = 5$ and $\lambda = 10$ requests per minute. We used these workloads because according to our tests under the available hardware and software configurations, higher workloads would exhaust the server resources. We use the open source Apache ODE process engine where an embedded Derby [16] database is used. The Amazon EC2 instance type is t1.micro with 1 vCPU and 0.594GiB memory. Each test run lasted for 60 minutes, but only the response times in the 30 minutes in the middle of this period have been considered (steady state).

The performance data is shown as Table II. Under the workload of $\lambda = 5$, the average performance overhead of our transformation mechanism is 92 ms. Under the workload of $\lambda = 10$, the average overhead is 130 ms. We conclude then that the performance overhead increases with the workload.

(a) Initiator (b) Responder

Figure 16. Correctness criteria of process transformation.



(a) Automaton model of the original process (b) Automaton model of the transformed process

Figure 17. Illustration of the correctness validation, robust *NWA* model of Figure 8.

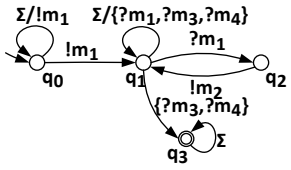


Figure 18. Correctness criteria for the illustrative process.

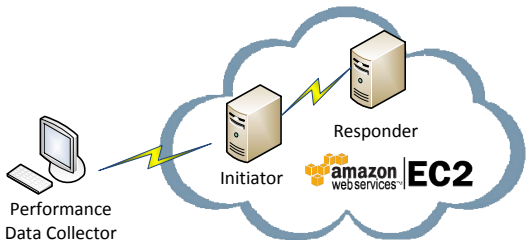


Figure 19. Setup of pperformance test.

TABLE II. PERFORMANCE OVERHEAD.

Origin	Trans	Overhead	Origin	Trans	Overhead
Workload $\lambda = 5$			Workload $\lambda = 10$		
287 ms	379 ms	92 ms	322 ms	452 ms	130 ms

However, we expect lower performance overhead when the infrastructure is scalable, like in a cloud environment.

C. Process Design Complexity

We have implemented the process designed in Figure 10, Figure 12 and Figure 13 using WS-BPEL. The number of activities before and after the process transformation is shown as Table III. The transformation is applied to one *reply* activity and the *receive* (or *pick*) activities in the set $next\_receive()$.

As an example, by applying our process transformations, a *reply* activity is replaced by one *sequence* structured activity and two basic activities: *reply* and *assign* nested in it.

```
<receive ...>
<assign name="assg1" ... />
<if ...>
  <condition ...>
  <assign name="assg2" ...>
  <else>
    <!-- Some Processing -->
    <assign name="assg3" ...>
  </else>
</if>
<reply ...>
```

V. Related Work

Solutions based on exception handling [17][18] is process-specific. WS-BPEL supports compensations of well-defined exceptions using exception handlers. However, elaborate process handler design requires process-specific knowledge of failure types and their related recover strategies. Alternatively, we try to ease the process designers from dealing with synchronization failures by a transparent process transformation from a given business process to its recovery-enabled counterpart.

A fault tolerant system can be built by coping with the occurrence of failures by applying redundancy [7]. Three kinds of redundancy are possible: information redundancy, time redundancy and physical redundancy. However, the existing solution either requires more effort of the business process designers, or additional infrastructure support, or both.

On physical layer, the robust solutions on process engine level [19][20] dependend on a specific process engine. We defined our solution based on the WS-BPEL building blocks without requiring extensions at the engine level. However, the

TABLE III. Number of activities before and after transformation.

	Before transformation	After transformation
Initiator	1 (*invoke*)	7 (2 *sequence*, 2 *assign*, 1 *while*, 1 *scope*, 1 *invoke*)
Responder	1 (*reply* in the set $next\_receive()$)	3 (1 *sequence*, 1 *reply*, 1 *assign*)
Responder	1 (*pick* with n *onMessage* branches in the set $next\_receive()$)	$2n + 5$ (1 *pick*, $n + 1$ *sequence*, 1 *reply*, $n + 1$ *assign*, 1 *while*)

transformed process can still be migrated to other standard process engines. Reliable network protocols such as HTTPR have been proposed to provide reliable synchronization. However, the deployment of these solutions increases the complexity of the network infrastructure. We assume that system crashes and network failures are rare events, thus extending the infrastructure may introduce too much overhead. Further, the solutions are not applicable in some outsourced deployment environments. For example, in some cloud computing environments, user-specific infrastructure configuration to enhance synchronization is not possible. Dynamic service substitution [21][22] is a way to perform recovery by replacing the target services by equivalent services. In [23][24], the QoS aspects of dynamic service substitution are considered. In our work, we do not change the business partners at runtime.

Information redundancy recovery is based on replication. Our cache-based process transformation is information redundant because a cache is a kind of replication. Time redundancy solutions include web services transactions. The WS-AT [25] standard specifies atomic web services transactions, while WS-BA [26] standard specifies relaxed transactions so that the participant can choose to leave the transaction before it commits. However, if a transaction rolls back, a process-specific compensation is required. Actually, transactions can deal with well-defined failures. The 2-phase commit distributed transaction protocol can not deal with system crash (referred to as *cite failure* in [27]). However, in a special case of process in which all participants send vote results to a coordinator, if the coordinator crashes before sending the vote results to any participant, all the participants are blocked and the final results of the transaction remain unknown.

VI. Conclusion and Future Work

In this paper, we have identified three types of interaction failures caused by system crashes and network failures and we have proposed a process interaction failure recovery method to cope with system crashes and network failures. This paper is an extension of our previous work [2][3][4][5], where we assumed that a certain pre-defined interaction follows the failed interaction. In this paper, we lift this limitation by allowing an arbitrary behavior to follow the failed interaction, making our solution more generally applicable. The challenge is to accept the resent message due to failures with the arbitrary control flow of the responder process. We transformed the business process design into nested word automata model. At a state that models the reception of an incoming message, we add an additional transition to accept the resent message due to failure. We have proved the correctness of our process transformations and we implemented a prototype to test the runtime performance of our method. The transformation complexity of our solution is analyzed. Currently, the transformation process is semi-automatic. We have implemented the automatic transformation from a WS-BPEL process to the NWA model, however, the transformation of the NWA model to the robust counter part is manually. In future, we will automate the transformation process and we will investigate more complex process interaction patterns.

References

[1] L. Wang, A. Wombacher, L. Ferreira Pires, M. J. van Sinderen, and C. Chi, "Robust interactions under system crashes and network failures of collaborative processes with arbitrary control flows," in The Seventh International Conferences on Advanced Service Computing, SERVICE COMPUTATION, 2015.

[2] L. Wang, A. Wombacher, L. Ferreira Pires, M. J. van Sinderen, and C.-H. Chi, "An illustrative recovery approach for stateful interaction failure of orchestrated processes," in IEEE 16th EDOC Workshops, 2012, pp. 38–41.

[3] ——, "A state synchronization mechanism for orchestrated processes," in IEEE 16th Intl. EDOC Conf., 2012, pp. 51–60.

[4] ——, "Robust client/server shared state interactions of collaborative process with system crash and network failures," in 10th IEEE Intl. Conf. on Services Computing (SCC), 2013.

[5] ——, "Robust collaborative process interactions under system crash and network failures," Intl. J. of Business Process Integration and Management, vol. 6, no. 4, 2013, pp. 326–340.

[6] A. Barros, M. Dumas, and A. Hofstede, "Service interaction patterns," in Business Process Management, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3649, pp. 302–318.

[7] A. S. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2006, ch. 8, pp. 321–375.

[8] Apache ODE, "Create a process," https://ode.apache.org/creating-a-process.html#in-memory-execution.

[9] SOA Technology for beginners and learners, "Transient vs. durable bpel processes," http://ofmxperts.blogspot.nl/2012/11/transient-vs-durable-bpel-processes.html, Nov. 2012.

[10] R. Alur and P. Madhusudan, "Adding nesting structure to words," J. ACM, vol. 56, no. 3, May 2009, pp. 16:1–16:43.

[11] OASIS, Web Services Business Process Execution Language, 2nd ed., OASIS, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, [retrieved: Feb., 2015], Apr. 2007.

[12] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow patterns," Distributed and Parallel Databases, vol. 14, no. 1, Jul. 2003, pp. 5–51.

[13] J. E. Hopcroft, Introduction to Automata Theory, Languages, and Computation, 3rd ed. Pearson Addison Wesley, 2007.

[14] J. Schiedung, "Analysing and modelling of IOTP transactions by CPNs and BPEL," Master's thesis, Darmstadt University of Technology, 2004.

[15] L. Wang, A. Wombacher, L. Ferreira Pires, M. J. van Sinderen, and C.-H. Chi, "A colloborative processes synchronization method with regards to system crashes and network failures," in the 29th Symp. on Applied Computing (SAC), 2014.

[16] Apache Software Foundation, "Ode database setup," http://ode.apache.org/databases.html.

[17] N. Russell, W. Aalst, and A. Hofstede, "Workflow exception patterns," in Advanced Information Systems Engineering, ser. Lecture Notes in Computer Science, E. Dubois and K. Pohl, Eds. Springer Berlin Heidelberg, 2006, vol. 4001, pp. 288–302.

[18] B. S. Lerner, S. Christov, L. J. Osterweil, R. Bendraou, U. Kannengiesser, and A. E. Wise, "Exception handling patterns for process modeling," IEEE Transactions on Software Engineering, vol. 36, no. 2, 2010, pp. 162–183.

[19] S. Modafferi, E. Mussi, and B. Pernici, "Sh-bpel: a self-healing plug-in for ws-bpel engines," in the 1st workshop on Middleware for Service Oriented Computing. NY, USA: ACM, 2006, pp. 48–53.

[20] A. Charfi, T. Dinkelaker, and M. Mezini, "A plug-in architecture for self-adaptive web service compositions," in IEEE Intl. Conf. on Web Services, Jul. 2009, pp. 35–42.

[21] M. Fredj, N. Georgantas, V. Issarny, and A. Zarras, "Dynamic service substitution in service-oriented architectures," in IEEE Congress on Services - Part I, Jul. 2008, pp. 101–104.

[22] L. Cavallaro, E. Nitto, and M. Pradella, "An automatic approach to enable replacement of conversational services," in Service-Oriented Computing, L. Baresi, C.-H. Chi, and J. Suzuki, Eds. Springer Berlin Heidelberg, 2009, vol. 5900, pp. 159–174.

[23] O. Moser, F. Rosenberg, and S. Dustdar, "Non-intrusive monitoring and service adaptation for ws-bpel," in the 17th intl. conf. on World Wide Web. NY, USA: ACM, 2008, pp. 815–824.

[24] F. Moo-Mena, J. Garcilazo-Ortiz, L. Basto-Diaz, F. Curi-Quintal, S. Medina-Peralta, and F. Alonzo-Canul, "A diagnosis module based on statistic and qos techniques for self-healing architectures supporting ws based applications," in Intl. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery, Oct. 2009, pp. 163 –169.

[25] OASIS Web Services Transaction (WS-TX) TC, Web Services Atomic Transaction (WS-AtomicTransaction), http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec.html, [retrieved: Feb., 2015], OASIS Standard, Rev. 1.2, Feb. 2009.

[26] ——, Web Services Business Activity (WS-BusinessActivity), http://docs.oasis-open.org/ws-tx/wstx-wsba-1.2-spec-os/wstx-wsba-1.2-spec-os.html, [retrieved: Feb., 2015], OASIS Standard, Rev. 1.2, Feb. 2009.

[27] M. T. Ozsu, Principles of Distributed Database Systems, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007, ch. 12.

How to Operate Container Clusters more Efficiently?

Some Insights Concerning Containers, Software-Defined-Networks, and their sometimes Counterintuitive Impact on Network Performance

Nane Kratzke, Peter-Christian Quint

Lübeck University of Applied Sciences
Department for Electrical Engineering and Computer Science
Center of Excellence CoSA
Germany
Email: {nane.kratzke, peter-christian.quint}@fh-luebeck.de

***Abstract*—In previous work, we concluded that container technologies and overlay networks typically have negative performance impacts, mainly due to an additional layer to networking. This is what everybody would expect, only the degree of impact might be questionable. These negative performance impacts can be accepted (if they stay moderate), due to a better flexibility and manageability of the resulting systems. However, we draw our conclusion only on data covering small core machine types. This extended work additionally analyzed the impact of various (high core) machine types of different public cloud service providers (Amazon Web Services, AWS and Google Compute Engine, GCE) and comes to a more differentiated view and some astonishing results for high core machine types. Our findings stand to reason that a simple and cost effective strategy is to operate container cluster with highly similar high core machine types (even across different cloud service providers). This strategy should cover major relevant and complex data transfer rate reducing effects of containers, container clusters and software-defined-networks appropriately.**

Keywords–Container; Cluster; Software-Defined-Network; SDN; HTTP; REST; Microservice; Overlay Network; Performance; Docker; Weave.

I. INTRODUCTION

In previous work [1], we investigated the performance impacts of container technologies like *Docker* [2] and overlay network solutions like *Weave* [3]. Both systems are typical type representants for container or overlay network solutions.

Container technologies and overlay network solutions are often mentioned in the same breath with the popular microservice approach [4]. The microservice architectural style is applied successfully by companies like Amazon, Netflix, or SoundCloud [4], [5]. This architecture pattern is used to build very large, complex and horizontally scalable applications composed of small, independent and highly decoupled processes communicating with each other using language-agnostic application programming interfaces (API). Therefore, microservice approaches and container-based operating system virtualization experience a renaissance in cloud computing. Especially, container-based virtualization approaches are often mentioned to be a high-performance alternative to hypervisors [6]. *Docker* [2] is such a container solution, and it is based on operating system virtualization. Recent performance studies show only little performance impacts to processing, memory, network or I/O [7]. That is why *Docker* proclaims itself a "lightweight virtualization platform" providing a standard runtime, image format, and build system for containers deployable to any Infrastructure as a Service (IaaS) environment.

Furthermore, container technologies and overlay network solutions are often used under the hood by popular container cluster platforms like *Mesos* [8], *CoreOS* [9] or *Kubernetes* [10]. The reader is referred to more detailed analysis of such kind of platforms [11], [12]. These cluster solutions are often the technological backbone of microservice based and highly scalable systems of cloud deployed systems. Nevertheless, corresponding performance impacts of the underlying technologies have been hardly investigated so far. Additionally, distributed cloud based microservice systems of typical complexity often use hypertext transfer protocol (HTTP) based and representational state transfer (REST) styled protocols to enable horizontally scalable system designs [13].

Beside this, container clusters often rely on so called overlay networks under the hood. Because overlay networks are often reduced to distributed hashtable (DHT) or peer-to-peer approaches, this paper uses the term software-defined-networks (SDN). SDNs, in the understanding of this paper, are used to provide a logical internet protocol (IP) network for containers on top of IaaS infrastructures.

This study investigates the performance impact of one representative SDN solution called *Weave* [3] for *Docker* [2] and it is outlined as follows. Section II presents related work about state-of-the-art network benchmarking of container and container cluster approaches and SDN solutions. If container clusters are to be distributed across multiple cloud infrastructures, the selection of appropriate machines is not trivial. Section III focuses on this problem and shows how and why we have selected exactly six virtual machine types of two different cloud service providers (AWS and GCE) to do our benchmarking. Section IV explains the experiment design to identify performance impacts of containers and SDNs. The used benchmark tooling is provided online [14]. Resulting performance impacts are analyzed and discussed in Section V. Derived conclusions and insights to minimize performance impacts on application, overlay network and IaaS infrastructure layer are presented in concluding Section VI.

II. RELATED WORK

To handle complexity of common microservice approaches, software developers often have to design, use or adapt existing SDN software, that will run on commodity hardware. One of the main challenges of SDN is to reach the performance known from proprietary software running on purpose-built hardware. This contribution will provide some insights into this gap, dealing with influences of popular container technologies, cluster approaches and software-defined-networks focusing HTTP and REST-based network performance often used in microservice architectures.

A. Container technologies

Although container based operating system virtualization is postulated to be a scalable and high-performance alternative to hypervisors. Hypervisors are still the standard approach for IaaS cloud computing [6]. Felter et al. provided a very detailed analysis on CPU, memory, storage and networking resources to explore the performance of traditional virtual machine deployments, and contrast them with the use of Linux containers provided via *Docker* [7]. Their results indicate that benchmarks that have been run in a *Docker* container show almost the same performance (floating point processing, memory transfers, network bandwidth and latencies, block I/O and database performances) like benchmarks run on "bare metal" systems. Nevertheless, Felter et al. did not analyze the impact of containers on top of hypervisors like this contribution does.

Furthermore, the reader should be aware that *Docker* is not the only container solution. Almost all operating systems provide some lightweight virtualization mechanisms like *Docker* does for the Linux and further operating systems. So to some degree, all of our conclusions can be transferred to lightweight virtualization approaches shown in Table I.

If more than one container has to be operated with **high availability and scalability requirements** in mind, so called container clusters come into play. We will discuss these technologies in Section II-B.

B. Container Cluster

Container clusters are similar to compute clusters. But container clusters run containers instead of processing jobs. It is up to the container cluster (to some definable degree) to decide which resource is allocated to run a container. Container clusters like *Mesos* [8], *CoreOS* [9] or *Kubernetes* [10] can be deployed on thousands of machines (nodes). These nodes can be hosted on "real hardware" machines but also on virtual machine instances in private or public clouds. A container cluster can be even deployed across different cloud service providers and different IaaS infrastructures. This results in some positive benefits for operation. (Regional) failures can be compensated, overload can be distributed and vendor lock-in can be avoided [11], [15]. However, all provided machines for such kind of cluster should show **similar performance characteristics** to enable fine-grained resource allocation capabilities of a container cluster [16]. This can be tricky to achieve in multi-provider scenarios and we will show in Section III how this was considered for our machine type selection.

Furthermore, container clusters often use SDN solutions under the hood to **hide networking specifics** from IaaS cloud service provider infrastructures. This will be discussed in following Section II-C.

C. Software-Defined-Networking

Software-defined-networking (SDN) is a paradigm where a software program dictates the overall network. SDN brings benefits, especially in cloud computing. Because it is easier to change and manipulate than using a fixed set of commands in proprietary network devices, SDN enables a high degree of flexibility. Cloud computing enables to launch or terminate instances very fast and on demand. In this context, accompanying scalability, elasticity and flexibility requirements are hard to handle with legacy network platforms [17].

Although there exist several SDN solutions for *Docker*. Pure virtual local area network (VLAN) solutions like *Open vSwitch (OVS)* [18] are not considered, because *OVS* is not to be designed for operating system virtualization. Two common SDN solutions are *Weave* [3] and *Flannel* [19]. *Weave* and *Flannel* are using a similar technique to build an overlay network. *Flannel* is developed by *CoreOS* [9] and optimized for the same named operating system. It turned out that *Flannel* can not be installed frictionless on another operating system without far-reaching modifications to the system. So we decided to use *Weave* as representative SDN solution. *Calico* [20] is a pure layer 3 approach to virtual networking for highly scalable data centers but was out of our focus because it requires *Docker 1.9* to work seamlessly. *Docker 1.9* had been released after our benchmark analysis. However, *Calico* is promising and will be extended to our here presented solution in ongoing work.

Weave creates a network bridge on *Docker* hosts to enable SDN for *Docker* containers. Each container on a host is connected to that bridge. A *Weave* router captures Ethernet packets from its bridge-connected interface in promiscuous mode. Captured packets are forwarded over the user datagram protocol (UDP) to *Weave* router peers running on other hosts. These UDP "connections" are duplex and can traverse firewalls.

D. Benchmarking network performances in containerized environments

To analyze the performance impact of containers, software-defined-networks and machine types, this paper considered several contributions on cloud related network performance analysis [21], [22], [23], [24], [25], [26], [27]. So far, there exist little papers focusing container technologies (only [7] provide some data). But none of these contributions focused explicitly on horizontally scalable systems with HTTP-based and REST-like protocols. To address this common use case for microservice architectures, this paper proposes a special experiment design being described in Section IV.

So, there exist almost no special network benchmarks focusing containerized environments and microservice related system design questions. But of course, there exist several well established **TCP/UDP networking benchmarks** like *iperf* [28], *uperf* [29], *netperf* [30] and so on. [24] provide a much more complete and comparative analysis of network benchmarking tools. [31] present a detailled list on cloud computing related (network) benchmarks. Most of these benchmarks focus pure TCP/UDP performance [24] and rely on one end on a specific server component used to generate network load. These benchmarks are valuable to compare principal network performance of different (cloud) infrastructures by comparing

TABLE I. Lightweight virtualization mechanisms on various operating systems

Operating system	Virtualization mechanism	Further links (last access 9th Nov. 2015)
Linux	Docker	http://www.docker.io
	LXC	http://linuxcontainers.org
	OpenVZ	http://openvz.org
Solaris	Solaris Zones	http://docs.oracle.com/cd/E26502_01/html/E29024/toc.html
FreeBSD	FreeBSD Jails	http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html
AIX	AIX Workload Partitions	http://www.ibm.com/developerworks/aix/library/au-workload/index.html
HP-UX	HP-UX Containers (SRP)	https://h20392.www2.hp.com/portal/swdepot/displayProductInfo.do?productNumber=HP-UX-SRP
Windows	Sandboxie	http://www.sandboxie.com
	Hyper-V Container (planned)	http://azure.microsoft.com/de-de/blog/microsoft-unveils-new-container-technologies-for-the-next-generation-cloud/

what maximum network performances can be expected for specific (cloud) infrastructures. But maximum expectable network performances are in most cases not very realistic for REST-based protocols.

Other **HTTP related benchmarks** like *httperf* [32] or *ab* [33] are obviously much more relevant for REST-based microservice approaches. These tools can benchmark arbitrary web applications. But because the applications under test are not under direct control of the benchmark, these tools can hardly define precise loads within a specific frame of interest. Therefore, HTTP related benchmarks are mainly used to run benchmarks against specific test resources (e.g., a HTML test page). But this makes it hard to identify trends or non-continuous network behavior.

Our presented approach is more a mix of above mentionend benchmarking strategies of pure TCP/UDP benchmarks and HTTP benchmarks. Section IV will show that we use a benchmark frontend (which is conceptually similar to *httperf* or *apachebench*) and a reference implementation under test (*ping-pong* system, which is conceptually similar to a *iperf* server but designed to measure HTTP performance). Furthermore, most of the above mentioned benchmarks concentrate on data measurement and do not provide appropriate visualizations of collected data. This may hide trends or even non-continuous network behavior. To cover this, our presented benchmarking approach focus data visualization as well. Often cloud service provider specific influences are not considered systematically in recent studies. To consider such cloud service provider specific influences on network performance, we performed a very systematic virtual machine type selection for our experiments (see Section III).

III. Virtual Machine Type Selection

Selecting virtual machine types can be complex in its details. The underlying decision making problem makes it difficult to model it appropriately. There exist several complex mathematical models like analytic hierarchy processes [34], utility function based methodologies [35], outranking approaches [36], simple additive weighting approaches [37], cloud feature models [38], dynamic programming approaches [39], cloud service indexing [40], ranking approaches [41], Markov decision process [42] or even astonishing combinations of Fuzzy logic, evidence theory and game theory [43] to support this task. However, in order to establish a container cluster across multiple providers, it is necessary to select most similar machines [16]. None of the above mentioned approaches focuses similarity. All mentioned approaches try to identify a best resource in terms of performance or cost effectiveness. Furthermore, very often cloud service selection is not done very systematically in practice. Often, the decision for a specific cloud service provider infrastructure is more an evolutionary process than a conscious selection. At some point in time, there might arise the need to change a cloud service provider or to diverse a deployment across several cloud service providers. There exist several technologies to support this. One approach is to use container cluster technologies like introduced in Section II-B. Well known companies like Google, Netflix, Twitter are doing this very successful to handle regional workloads, to provide failover and overflow capacities. Small and medium sized companies can benefit as well. Nevertheless, these type of clusters rely on homogeneous nodes (nodes with similar performance characteristics). Otherwise, the intended fine-grained resource allocation of container clusters gets limited.

EasyCompare [31] is a tool with the focus on identifying most similar (not best performing or most cost effective) machine types across different IaaS cloud service providers. The suite uses a feature vector shown in equation (1) to describe the performance of a virtual machine type. EasyCompare uses several different benchmarks. **Processing performance** is determined by the **Taychon** benchmark [44]. The **Stream** benchmark is used to measure the **memory performance** [45]. With the **IOZone benchmark** [46] the read and write **disk performance** can be measured. iperf [28] is used to measure **network transfer rates** of intra cloud data transfers.

$$\mathbf{i} = \begin{pmatrix} i_1 \\ i_2 \\ \\ i_3 \\ i_4 \\ \\ i_5 \\ \\ i_6 \\ \\ i_7 \\ \end{pmatrix} \begin{array}{l} \text{Processing: Amount of simultaneous executable threads} \\ \text{Processing: Processing time in seconds} \\ \text{(Median of all Tachyon benchmark runs)} \\ \text{Memory: Memory size in MB} \\ \text{Memory: Memory transfer in MB/s} \\ \text{(Median of all Stream Triad benchmark runs)} \\ \text{Disk: Data transfer in MB/s for disk reads} \\ \text{(Median of all IOZone read stress benchmark runs)} \\ \text{Disk: Data transfer in MB/s for disk writes} \\ \text{(Median of all IOZone write stress benchmark runs)} \\ \text{Network: Data transfer in MB/s via network} \\ \text{(Median of all iperf benchmark runs)} \end{array} \quad (1)$$

The similarity $s(\mathbf{i}, \mathbf{j})$ of two virtual machine types $\mathbf{i}$ and $\mathbf{j}$ can be analyzed by calculating their vector similarity. Although this approach is mostly used in domains like information retrieval, text and data mining, vector similarities can be used to determine the similarity of virtual machine types as well. A

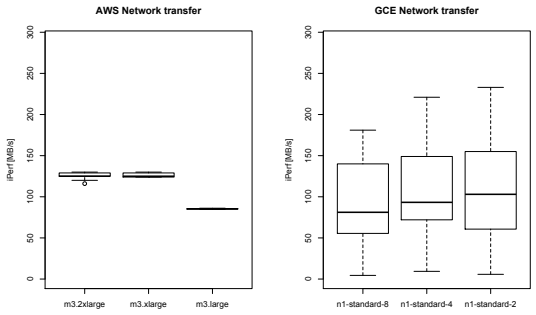


Figure 1. Boxplots of measured network performances using iperf on selected virtual machine types, taken from [31]

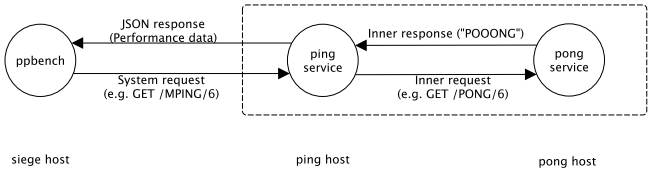
good similarity measure has to fulfill three similarity intuitions [47]:

- **Intuition 1:** "*The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.*"
- **Intuition 2:** "*The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.*"
- **Intuition 3:** "*The maximum similarity between A and B is reached when A and B are identical.*"

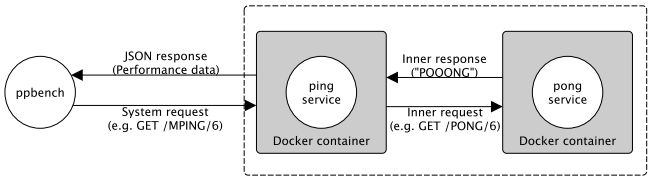
According to [31], it turned out that the normalized euclidian distance based shown in equation (2) measure fulfills Lin's intuitions [47] of a good similarity measure for the intended purpose of identifying most similar cloud resources across different providers. This variant of the Euclidian distance metric measures the relative performance relations of performance components $(i_1, i_2, \ldots, i_m)$ and $(j_1, j_2, \ldots, j_m)$ and normalizes the result $s(\mathbf{i}, \mathbf{j})$ to a similarity value between 0.0 ($\mathbf{i}$ and $\mathbf{j}$ are not similar at all) and 1.0 ($\mathbf{i}$ and $\mathbf{j}$ are completely similar).

$$
\begin{aligned}
\forall \mathbf{i}, \mathbf{j} \qquad & s(\mathbf{i}, \mathbf{j}) = 1 - \frac{1}{m} \sum_{n=1}^{m} \left(1 - \frac{\min(i_n, j_n)}{\max(i_n, j_n)}\right)^2 \\
\forall \mathbf{i}, \mathbf{j} \qquad & s(\mathbf{i}, \mathbf{j}) = s(\mathbf{j}, \mathbf{i}) \\
\forall \mathbf{i}, \mathbf{j} \qquad & 0 \leq s(\mathbf{i}, \mathbf{j}) \leq 1 \\
\forall \mathbf{i} \qquad & s(\mathbf{i}, \mathbf{i}) = 1
\end{aligned}
\tag{2}
$$

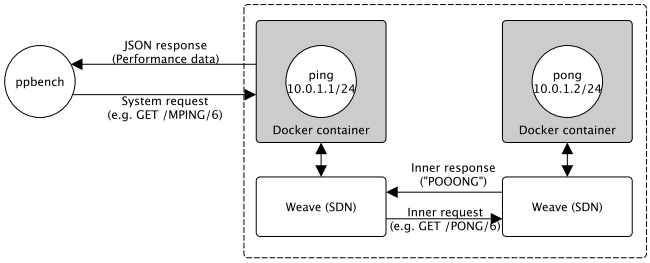
The reader is referred to Appendix Table IV. This compares machine types provided by two major and representative public cloud service providers: Amazon Web Services and Google Compute Engine. Finally, in over 500 different possible counterpart pairings (see Table IV) only three machine pairings have a strong similarity of almost 1. These pairings are reasonable choices for container clusters, and that is why we chose them as objects of investigation. So the AWS instance types m3.2xlarge, m3.xlarge, m3.large and GCE machine types n1-standard-8, n1-standard-4 and n1-standard-2 are used for the determination of performance impacts when using *Docker* and SDN technologies. Their measured network performance is shown in Figure 1.



(a) **Bare** experiment to identify reference performance



(b) **Docker** experiment to identify impact of container



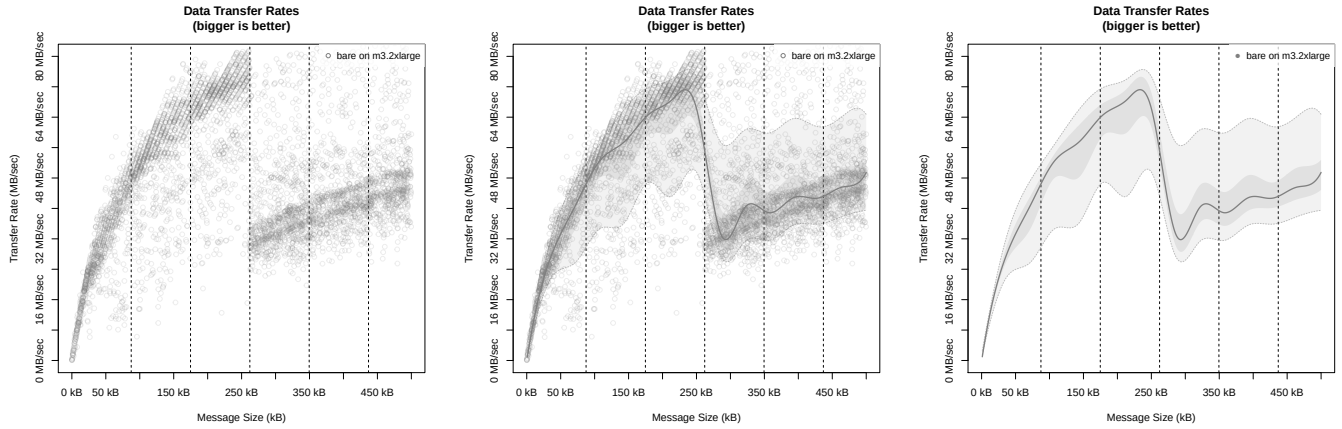
(c) **Weave** experiment to identify impact of SDN

Figure 2. Experiment setups

IV. EXPERIMENT DESIGN

To analyze the network performance impact of container, SDN layers and machine types on the performance impact of distributed cloud based systems using HTTP-based REST-based protocols, several experiments have been designed (see Figure 2). The analyzed *ping-pong* system provides a REST-like and HTTP-based protocol to exchange data. This kind of connections are commonly used in microservice architectures and container based cloud systems. The *ping* and *pong* services were developed using Googles *Dart* programming language [48]. It is possible for the siege to send a request to the *ping-pong* system. Via this request the inner message length between *pong* and *ping* server can be defined. So, it is possible to measure HTTP based transfer rates and answer times between *ping* and *pong* for a specific message size.

In our initial contribution [1] we installed *Apachebench* on the siege server [49] to collect performance data of the *ping-pong* system. But with high-core virtual machines we identified several network connection problems, which almost did not occur on low core machines like (AWS m3.medium and m3.large, GCE n1-standard-2). Especially in the range of message sizes of about 250kB we identified a lot of network connection errors between *ping* and *pong* host on high core machines (the effect was less severe or did not occur at all with smaller and bigger message sizes). While



(a) Plot of measurements

(b) Plot of measurements with median line and 90% and 50% confidence intervals

(c) Plot of median line and confidence intervals only, still enables to estimate skewness of distribution while reducing jitter in presentation

Figure 3. Different visualizations of the same data by example transfer rates measured on a m3.2xlarge instance

it was not clear of first, what causes these phenomenon, it turned out that this was due to flooding connection requests at a critical point that relates with the standard TCP receive window size configured on *ping* and *pong* host (see Section V-B for analysis). This effect did not occur with single (like m3.medium[1]) or even double core machines. But on high core machines *Apachebench* pushed the system into oversaturated network conditions producing useless benchmark data. That is why we decided to adapt our experiment setting to some degree.

Instead of *Apachebench* we decided to run our experiments with a special developed benchmarking script (*ppbench*). *Ppbench* sends several HTTP request with a random message size m between a minimum and maximum amount of bytes to the *ping* server to analyze the answer and response behaviour for different message sizes. For this contribution we covered the message sizes between 1 byte and 500kB. *Ppbench* was used to collect a 10% random sample of all possible message sizes. The *ping* server relays each request to the *pong* server. And the *pong* server answers the request with a m byte long answer message (as requested by the HTTP request). The *ping* server measures the time from request entry to the point in time when the *pong* server answer hit the *ping* server again. If there are connection problems the *ping* server retries to connect the *pong* server for a specified amount of times. If the *ping* server can not connect with the *pong* server it answers with a HTTP 503 code. The *ping* server answers the request with the measured time between *ping* and *pong*, the length of the returned message, the HTTP status code send by *pong* and the number of retries to establish a HTTP connection between *ping* and *pong*. *Ppbench* repeats a request for a specified message size m several times (in our case 10 times) to calculate a mean transfer rate. Using this approach we got a much more complete coverage and insight into our analyzed problem domain.

The **Bare experiment setup** shown in Figure 2(a) was used to collect reference performance data of the *ping-pong* system deployed to different virtual machines interacting with a REST-like and HTTP based protocol. Further experiments added a container and a SDN solution to measure their impact on network performance. A *ping* host interacts with a *pong* host to provide its service. Whenever the *ping* host is requested by *siege* host, the *ping* host relays the original request to the *pong* host. The *pong* host answers the request with a response message.

The intent of the **Docker experiment setup** was to figure out the impact of an additional container layer to network performance (Figure 2(b)). So, the *ping* and *pong* services are provided as containers to add an additional container layer to the reference experiment. Every performance impact must be due to this container layer.

The intent of the **Weave experiment setup** shown in Figure 2(c) was to figure out the impact of an additional SDN layer to network performance. *Weave*, a representative SDN solution, connects the *ping* and the *pong* containers. So, every data transfer must pass *Weave* between *ping* and *pong*. Therefore, every performance impact must be due to this additional networking layer.

V. DISCUSSION OF RESULTS

To generate statistical significant data, our benchmarking tool generates for the analyzed space of message sizes (1 byte upto 500kB) something about 5000 data points. Each benchmarked data point aggregates 10 single measurements. And this was done for six machine types of two providers several times. In other words, we have a lot of data points to present (and to confuse the reader). We will explain our presentation of this data in Section V-A. Section V-B describes an astonishing non-continuous behavior of data transfer rates, which will guide us contemplating about container impact in Section V-C and SDN impact in Section V-D.

A. Presentation of data

All relevant statistical data processing and data presentation has been done by statistical computing toolsuite R [50]. We decided to present the data per machine type as transfer plots

[1]This was the machine type we used for our initial publication [1]

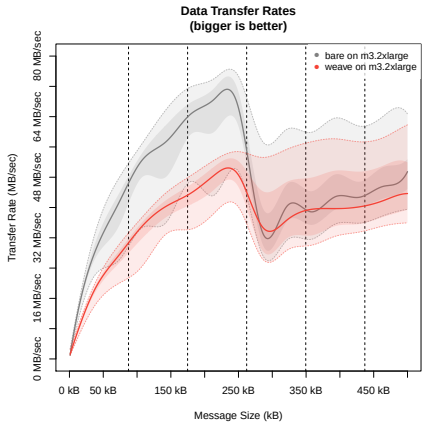
(to visualize the changing data transfer rates of different message sizes) and loss plots (to visualize the relative performance change of a data series compared with a reference data series). All transfer plots can be found in Figure 5, all loss plots can be found in Figure 6. In the following subsections we explain how these plots have been generated and how these plots have to be read by the reader.

1) Transfer plots: If we would barely plot our measurements, we would get a graphical result like in Figure 3(a). This kind of presentation provides a good and realistic impression about the distribution of measurements but without guiding descriptive statistical data. To draw general conclusions, we should not be interested in single data points but in trends and confidence intervals. Figure 3(b) shows exactly the same dataset but with a (spline smoothed) median line and 50% (indicating all values between the 25% and 75% percentile) and 90% confidence intervals (indicating the range of all measured values between the 5% and 95% percentile) as an overlay. This helps to reason about the skewness of a distribution and to handle outliers statistical appropriate. So Figure 3(b) provides the most information. But if we compare two or three series of (partly overlaying) data, it should be obvious for the reader that to plot all data points would mean a lot of confusion. The reader would get lost in details of thousands of data points. That is why we have decided to plot only the descriptive statistical data (median, 50% and 90% confidence interval) if we compare two or more data series. However, the reader should be aware that the reduced presentation of data – like done exemplarily in Figure 3(c) – still is based on a statistical significant amount of data points like shown in Figure 3(a). This is true for all data presentations shown in Figure 5. But for a better readability we decided not to plot the data points in comparison plots.

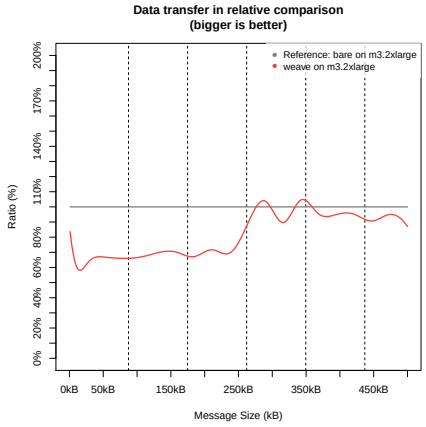
2) Loss plots: Transfer plots as shown in Figure 4(a) enable us to compare two or more data series by looking at their descriptive statistics of their absolute values. But to compare data series in a relative way is more useful, especially if we consider that machine types can be provided by different cloud services providers (in our case AWS and GCE). The reader may consider Figure 1 to see that AWS and GCE provide similar but not identical data transfer rates in their infrastructures for specific virtual machine types. Even the variances may differ substantially; AWS infrastructure shows almost no variances (very small box plots), but GCE shows substantial variances in data transfers). So, it is not helpful to compare absolute values to reason about performance impacts of container and SDN solutions in different IaaS infrastructures and for different virtual machine types due to different absolute levels of network performances and differing variances.

Therefore, we provide additional plots (see Figure 4(b)), to visualize the relative performance impact of containers or SDN compared with a reference experiment (in our case this is always the bare experiment on a specific machine type). A loss line is simply the division of two median lines (the two median lines $red/grey$ shown in Figure 4(a)).

$$\begin{aligned} loss_{exp,mach}(m) &= \frac{trans_{mach,exp}(m)}{trans_{mach,\text{bare}}(m)} \\ mach &\in \{\text{m3.large}, ..., \text{n1-standard-8}\} \\ exp &\in \{\text{bare}, \text{docker}, \text{weave}\} \end{aligned} \tag{3}$$



(a) Exemplary absolute comparison of two data series



(b) Exemplary relative comparison of two data series (same data set as above)

Figure 4. Absolute and relative comparison of two datasets by example of m3.2xlarge bare and *Weave* experiment data

And of course the loss of $loss_{\text{bare},mach}(m) = 1$. That is the grey reference line in Figure 4(b). All loss plots of all analyzed machine types are presented in Figure 6.

B. TCP receive windows define small and big messages

While transfer plots and loss plots are helpful to identify general trends, it is worth to come back to the bare plots like shown in Figure 3(a) exemplarily. The reader can identify a sharp break of transfer rates at about 250kB. This is exactly the same range where *Apachebench* produced a lot of network errors for high core machine types. We can measure transfer rates up to a specific transfer rate for messages smaller 250 kB. But these transfer rates are almost halfed abruptly for message sizes greater than 250 kB for a substantial amount of measurements (but not for all). It turned out that this effect could be measured for all analyzed machine types (see Appendix Figure 7) and in both IaaS infrastructures (AWS, GCE). So, it is not due to some traffic shaping caused by the IaaS infrastructure. Furthermore, the effect occurred exactly at three times the standard TCP receive window size (87380 bytes) of the used Ubuntu Linux 14.04 systems under test. The reader is referred to RFC 1323/7323 [51], [52] if being

unfamiliar with TCP window sizes. That is why we plot all multiples of the TCP standard receive window in our plots (dashed lines) to provide the reader some visual guidance of this effect.

We are quite sure that this has to do with the TCP protocol stack and/or a combination with the used *Dart* language and its HTTP library. For investigation and cross-checking we implemented the *ping-pong* system in *Java* as well and used the standard *Java* HTTP library (com.sun.net.httpserver). It turned out that the *Java* implementation of the *ping-pong* system does not show exactly the same effect but something similar. The *Java* curve reached its peak (and stayed on that level) at about three times the TCP standard receive window size, it stays below the *Dart* curve in the second and third TCP standard receive window but showed slightly better results in the first, fourth and fifth receive window. Finally, in the six window it showed the same performance as the *Dart* implementation. Furthermore, the *Java* implementation showed some significant outliers to the bottom exactly around the TCP standard receive window. In total (from first to sixth TCP standard receive window) the *Dart* implementation showed better results than the *Java* implementation. So Google seemed to optimized its *Dart* HTTP library at the cost of non-linear behavior. So, whatever the cause for this effect is, it seems have something to do with the TCP stack. However, the effect enables us to define a clear criterion to define small and big message sizes (for *Dart* and other languages as well). And the reader will see that containers and SDNs behave different for small and big messages sizes.

> Thus, we define **small messages** and **big messages** as messages smaller/bigger than three times (for *Dart*, for *Java* and other languages this would be another value) the **TCP standard receive window size** of a system.

In other words, small and big messages are operating system and virtual machine specific but configurable. The TCP standard receive window size is configurable on Linux systems for instance.

C. Container impact

The container impact for all machines can be seen in Figure 6 (all red lines). Table II summarizes these figures to some handsome guidance levels (and, therefore, not exact numbers). We see that the container impact is responsible to reduce the network transfer rates down to about 90% for small message sizes. But we although notice an astonishing effect. Transfer rates are **increasing** for big message sizes. The transfer rates are increased up to 120% to 130% for 4 and 8 core machines. That means an additional layer improves the overall data transfer performance (for big messages). This effect is hardly measurable for small core machines. Containers enlarge the network stack and therefore introduce more buffering capacities. This is obviously of minor relevance for small messages but seems to be positive for big messages.

D. SDN impact

The impact of SDNs for all machines can be seen in Figure 6 (all blue lines). Table III summarizes these figures to some handsome guidance levels (and, therefore, not exact numbers). The SDN impact can be much more severe than the container impact. This is especially true for small core virtual machine types like m3.large (AWS), n1-standard-2 (GCE). This has to do with the fact that SDN software routers contend for the same (and very limited) CPU with processing services. The worst case we have measured is for the n1-standard-2 GCE virtual machine type, where this effect is responsible to reduce the network transfer rates to about 25% of the bare reference system.

For 4-core machine types this effect is getting minimized and transfer rates are going down to about 60% of the bare reference system. For 8-core machine types this loss can be reduced to 70% of the bare reference system for small messages.

And again we see an astonishing effect for big messages. Transfer rates are **recovering** for big message sizes. On the AWS infrastructure the data transfer performance of m3.xlarge and m3.2xlarge instances is hardly distinguishable from the bare reference experiment for big messages. The GCE infrastructure seems to be more susceptible and is recovering less extensive for big messages. However, it is recovering! This might have to do with a general higher data transfer variance, which can be observed in the GCE infrastructure (see Figure 1). Like containers, SDNs enlarge the network stack and, therefore, introducing much more buffering capacities. This results in negative performance impacts for small messages but seems to be positive for big messages on high core virtual machine types.

E. Summary

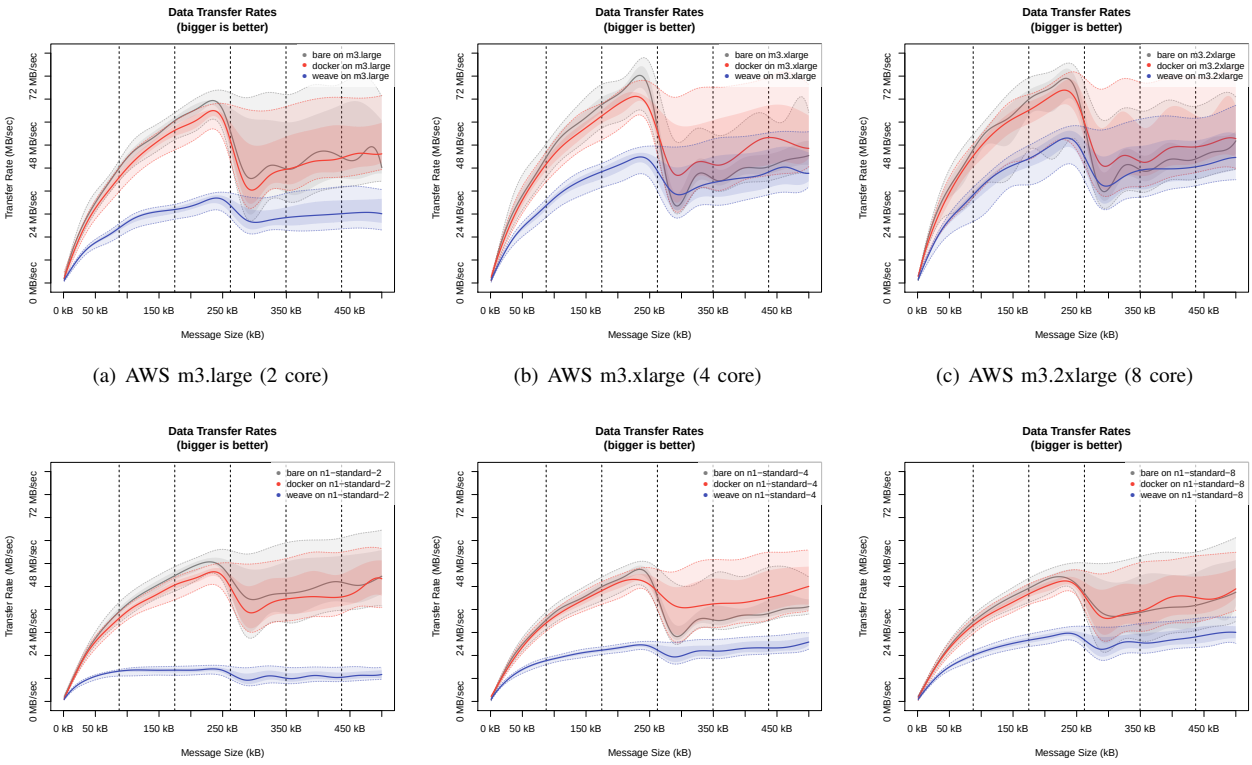
In our initial publication, we concluded that container and SDN solutions reduce network performance due to enlarging network stacks (this effect could be measured on single core machines very well). But these solutions are inducing more buffering capacities along an extended network stack as well. So, containers and SDNs are accompanied by positive and negative effects at the same time. In fact, it turned out, that an additional container layer can even improve the overall network performance for high core machine types. SDN solutions can have severe impacts. This is especially true for low core virtual machine types. However, on high core machines, the impact is moderate and for big message sizes SDN solutions might be even hardly measurable in data transfer rates (see Figure 6(c,e)).

VI. CONCLUSION AND OUTLOOK

To use HTTP-based, REST-like containerized services is a common approach for modern horizontally scalable and cloud

TABLE II. Relative performance impacts of an **additional container layer** (*Docker*) to data transfer rates; just guidance levels; please check Figure 6 (red lines) for more details

Machine type	cores	Small messages	Big messages
AWS m3.large	2	90% - 100%	100% - 110%
GCE n1-standard-2	2	95%	95% - 100%
AWS m3.xlarge	4	90%	120% - 130%
GCE n1-standard-4	4	95%	120% - 130%
AWS m3.2xlarge	8	90%	110% - 130%
GCE n1-standard-8	8	90%	110% - 120%



(a) AWS m3.large (2 core)

(b) AWS m3.xlarge (4 core)

(c) AWS m3.2xlarge (8 core)

(d) GCE n1-standard-2 (2 core)

(e) GCE n1-standard-4 (4 core)

(f) GCE n1-standard-8 (8 core)

Figure 5. Median transfer rates with 90% confidence bands (5% and 95% percentile) and TCP standard receive window of 83780 bytes (dotted lines)

TABLE III. Relative performance impacts of an **additional SDN layer** (*Weave*) to data transfer rates; just guidance levels; please check Figure 6 (blue lines) for more details

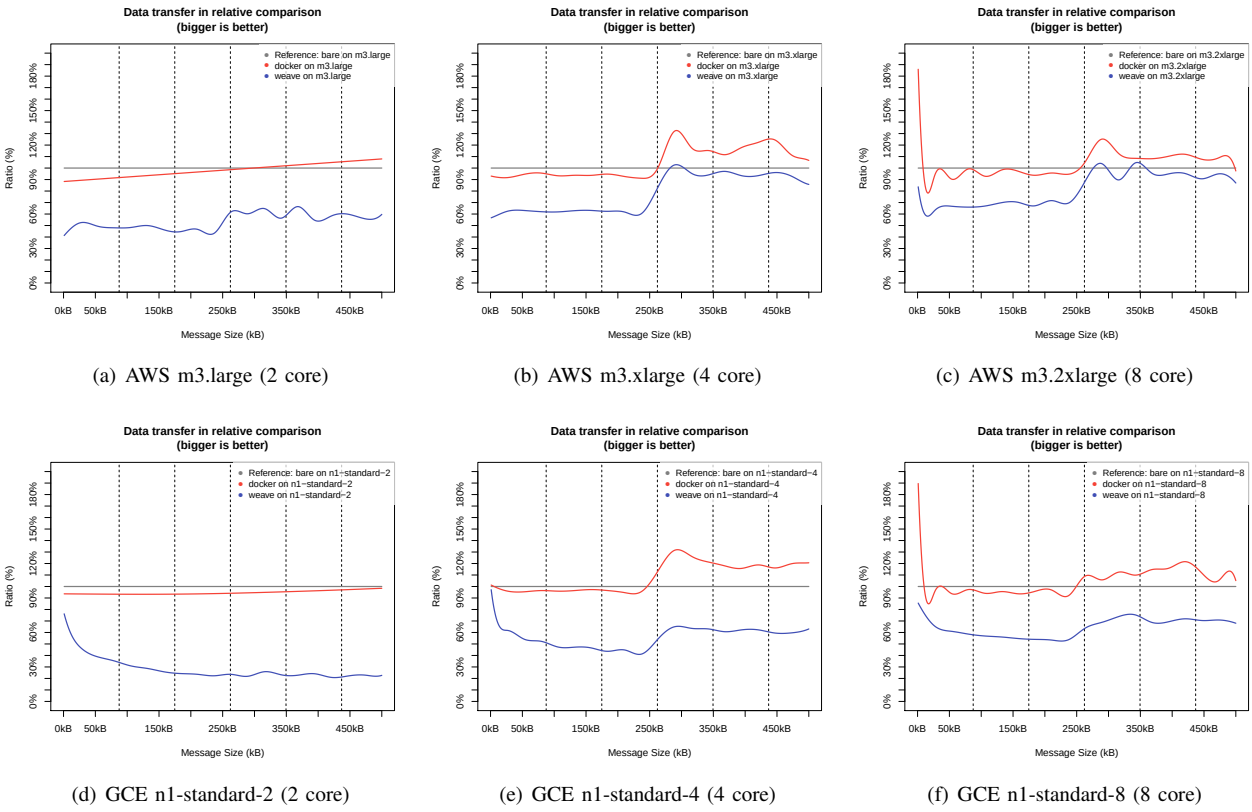
Machine type	cores	Small messages	Big messages
AWS m3.large	2	45%	60%
GCE n1-standard-2	2	25% - 80%	25%
AWS m3.xlarge	4	60%	90% - 100%
GCE n1-standard-4	4	40% - 100%	60%
AWS m3.2xlarge	8	70% - 80%	90% - 100%
GCE n1-standard-8	8	55% - 80%	70%

deployed microservice based systems. These technologies are not intentionally used to make cloud systems more performant but to make their operations and deployment more manageable. Astonishingly, we identified several cases where these technologies could even improve data transfer performances.

The impact of containers and SDNs on data transfer rates seems related with TCP standard receive windows (see RFC 1323/7323 [51], [52]). The window size allows a precise criterion what small and big messages are. Furthermore, we identified that small and big messages can show substantial differing transfer rates. This insight might be valuable for network optimizations of container clusters due to the fact, that TCP standard receive windows sizes can be configured by an operating system. However, this point stays open for ongoing research. In ongoing research we investigate further programming languages like Go, Ruby and Java to cover more programming languages and language specific behaviors because we identified that non-continuous network behavior seems to be deeply language (or library) specific. According to our preliminary results, almost every analyzed programming language (or their standard HTTP libraries) showed such non-continuous points.

For five of six analyzed machine types from AWS and GCE, we even identified a positive effect of containers on data transfer rates for big messages (see Table II) and we even identified a recovering effect on transfer rates for big messages due to SDN (see Table III). Furthermore, it is interesting to know that the performance loss of SDN for small messages is much more intensive on low core machines (in worst case on a GCE n1-standard-2 machine we only measured 25% of the reference performance) than on high core machines (about 80% of the reference performance).

So far, the presented data has some shortcomings. We only presented data transfer rates, but the used benchmarking solution is able to generate other metrics like round trip latencies or requests per second as well. However, the derived insights are basically the same. So, we did not present these metrics here. Our systematic virtual machine type selection has been only applied to AWS and GCE but can be easily extended to other public cloud service infrastructures like Microsoft Azure or private cloud infrastructures like *OpenStack* or *Eucalyptus*. This is up for ongoing work. For pure pragmatic reasons, we only covered message sizes up to 500kB. Modern NoSQL databases can easily return much larger messages. However, we pushed our system under test into a saturated network condition. So we think, it is unlikely to get new insights



(a) AWS m3.large (2 core)

(b) AWS m3.xlarge (4 core)

(c) AWS m3.2xlarge (8 core)

(d) GCE n1-standard-2 (2 core)

(e) GCE n1-standard-4 (4 core)

(f) GCE n1-standard-8 (8 core)

Figure 6. Relative transfer rates and TCP standard receive window of 83780 bytes (dotted lines)
$< 100\%$ means performance loss, 100% means no loss, $> 100\%$ means performance improvement

beyond 500kB messages. But of course, it might be possible that there exist further non-continuous network behavior points like we have seen for the Dart implementation of the *ping-pong* system. Furthermore, we only covered one SDN solution so far. Our ongoing efforts concentrate on extending the benchmarking solution *ppbench* by additional overlay networks like *Flannel* [19] (often used for *Kubernetes*), *Calico* [20] (a pure layer 3 approach to virtual networking for highly scalable data centers) or the recently released built-in overlay network feature of *Docker 1.9 (libnetwork)* to harden and eliminate possible solution specific conclusions.

So far, all of our study results indicate that our collected data can be used to do a systematic virtual machine type selection for cloud deployed HTTP-based and REST-like systems of general applicability. This is especially true for microservice based systems being deployed on container clusters like *Mesos*, *Kubernetes* or *Docker Swarm*.

It seems to be a simple and cost effective strategy to operate container clusters with **most similar high core machine types** across different providers.

This strategy should cover all relevant and complex performance related implications of containers, container clusters and SDNs appropriately. So, container and SDN technologies must not be avoided in general due to performance apprehensions. Under special circumstances and deployed on high core machine types, containers can even show better performances! Furthermore, container and SDN technologies provide more flexibility and manageability in designing complex horizontally scalable distributed cloud systems. But they should be always used with the above mentioned performance implications in mind.

Taking all together, a similarity-based virtual machine type selection can be used easily to optimize network performance for container cluster based cloud computing. Finally, the reader might be pleased to hear, that the selected machine types for this contribution fulfilled the criterion to be the most similar machine types across AWS and GCE, but these selected machine types are far from the most expensive machine types provided by both providers. Thus, a similarity-based virtual machine type selection seems to be a cost effective strategy as well.

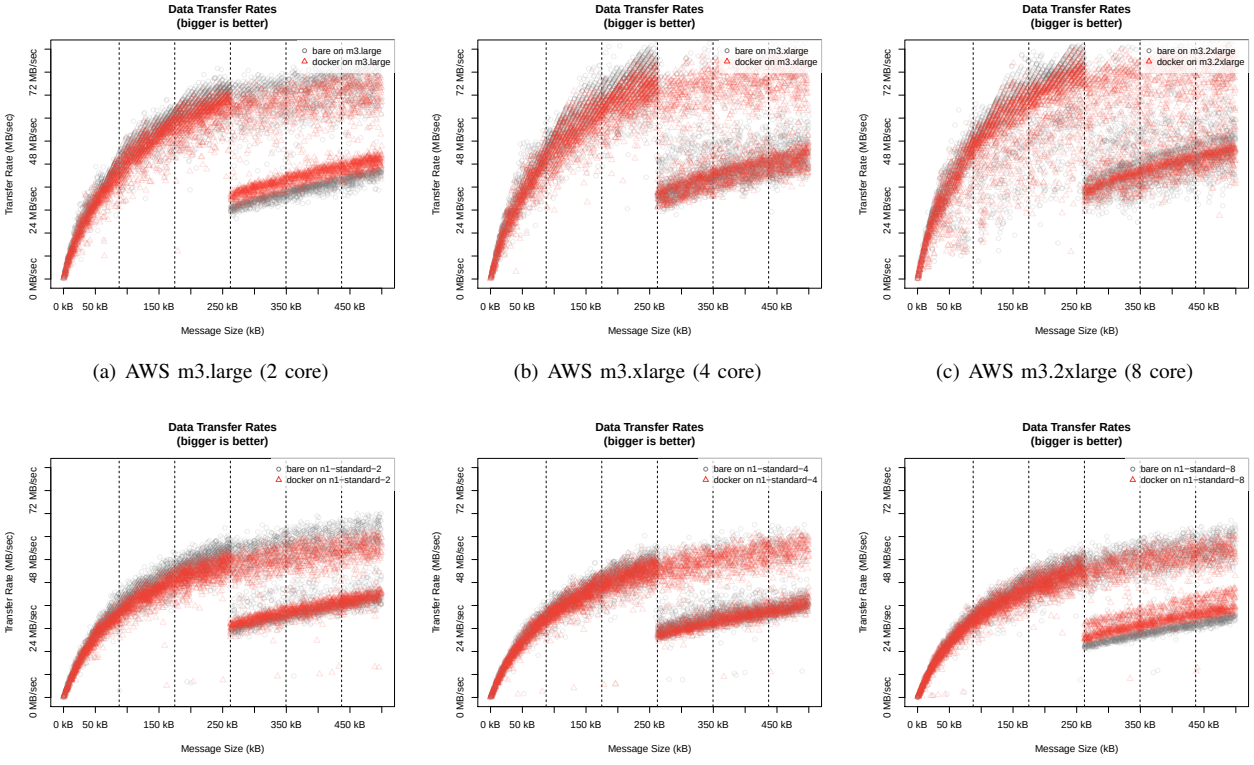
ACKNOWLEDGMENT

This study was funded by German Federal Ministry of Education and Research (Project Cloud TRANSIT, 03FH021PX4). We used research grants by courtesy of Amazon Web Services and Google Compute Engine. We also thank Bryan Boreham of Weaveworks for checking our *Weave* data for our initial conference paper [1]. We tried to realize his valuable insights and suggestions in this extended version. And finally, the authors thank Lübeck University (Institute of Telematics) and fat IT solution GmbH (Kiel) for their support of Cloud TRANSIT.

APPENDIX

TABLE IV. Similarity of AWS and GCE virtual machines types (sorted on both axis by descending amount of simultaneous executable threads).

Similarities	n1-highmem-32	n1-standard-32	n1-highcpu-32	n1-highmem-16	n1-standard-16	n1-highcpu-16	n1-highmem-8	n1-standard-8	n1-highcpu-8	n1-highmem-4	n1-standard-4	n1-highcpu-4	n1-highmem-2	n1-standard-2	n1-highcpu-2	n1-standard-1	g1-small	f1-micro
m4.10xlarge	0.67	0.70	0.68	0.56	0.50	0.52	0.51	0.46	0.48	0.40	0.41	0.43	0.36	0.37	0.40	0.35	0.30	0.13
c4.8xlarge	0.66	0.66	0.70	0.59	0.65	0.54	0.55	0.47	0.50	0.41	0.43	0.46	0.37	0.39	0.43	0.38	0.32	0.14
c3.8xlarge	0.81	0.82	0.81	0.56	0.64	0.51	0.52	0.45	0.47	0.39	0.40	0.42	0.34	0.35	0.39	0.33	0.27	0.13
i2.4xlarge	0.61	0.57	0.60	0.84	0.77	0.80	0.66	0.60	0.64	0.52	0.54	0.59	0.46	0.48	0.53	0.47	0.40	0.22
r3.4xlarge	0.60	0.57	0.60	0.83	0.76	0.79	0.65	0.59	0.62	0.50	0.52	0.56	0.44	0.45	0.51	0.44	0.37	0.21
m4.4xlarge	0.61	0.58	0.61	0.84	0.77	0.80	0.66	0.60	0.64	0.51	0.54	0.58	0.45	0.47	0.52	0.46	0.39	0.21
c3.4xlarge	0.60	0.57	0.60	0.83	0.76	0.79	0.65	0.72	0.62	0.50	0.52	0.56	0.43	0.45	0.50	0.44	0.37	0.21
c4.4xlarge	0.59	0.55	0.58	0.82	0.75	0.79	0.64	0.72	0.62	0.53	0.56	0.60	0.49	0.51	0.56	0.50	0.44	0.25
i2.2xlarge	0.52	0.49	0.51	0.62	0.63	0.65	0.79	0.81	0.83	0.60	0.62	0.63	0.52	0.53	0.54	0.51	0.43	0.27
r3.2xlarge	0.52	0.49	0.51	0.62	0.63	0.65	0.79	0.81	0.83	0.60	0.62	0.63	0.52	0.53	0.54	0.51	0.43	0.27
m4.2xlarge	0.46	0.43	0.45	0.55	0.57	0.60	0.74	0.77	0.78	0.57	0.60	0.60	0.49	0.51	0.50	0.53	0.54	0.37
m3.2xlarge	0.52	0.49	0.50	0.61	0.63	0.66	0.79	0.96	0.83	0.60	0.63	0.63	0.52	0.54	0.54	0.52	0.44	0.27
c3.2xlarge	0.52	0.50	0.51	0.62	0.64	0.80	0.79	0.82	0.83	0.60	0.76	0.62	0.52	0.53	0.53	0.51	0.44	0.27
c4.2xlarge	0.53	0.51	0.52	0.62	0.64	0.80	0.78	0.81	0.82	0.58	0.74	0.60	0.50	0.52	0.52	0.50	0.42	0.26
i2.xlarge	0.45	0.42	0.43	0.54	0.59	0.59	0.58	0.78	0.62	0.81	0.82	0.80	0.60	0.60	0.58	0.57	0.48	0.32
r3.xlarge	0.46	0.42	0.43	0.54	0.59	0.60	0.58	0.79	0.63	0.82	0.83	0.81	0.62	0.62	0.60	0.58	0.50	0.33
m4.xlarge	0.47	0.44	0.45	0.56	0.61	0.61	0.60	0.66	0.64	0.83	0.85	0.82	0.63	0.63	0.60	0.60	0.51	0.34
m3.xlarge	0.47	0.43	0.45	0.56	0.58	0.74	0.60	0.63	0.64	0.80	0.96	0.82	0.59	0.61	0.61	0.58	0.49	0.31
c3.xlarge	0.46	0.42	0.43	0.54	0.60	0.59	0.58	0.65	0.77	0.82	0.83	0.79	0.61	0.74	0.57	0.56	0.48	0.32
c4.xlarge	0.49	0.46	0.47	0.57	0.62	0.62	0.61	0.67	0.79	0.81	0.83	0.80	0.59	0.74	0.57	0.57	0.48	0.32
r3.large	0.39	0.36	0.37	0.45	0.54	0.65	0.48	0.56	0.52	0.64	0.76	0.58	0.83	0.81	0.77	0.65	0.57	0.40
m3.large	0.41	0.38	0.39	0.48	0.54	0.53	0.51	0.58	0.69	0.63	0.64	0.61	0.84	0.98	0.80	0.67	0.58	0.39
c3.large	0.39	0.37	0.37	0.46	0.54	0.51	0.49	0.57	0.52	0.64	0.63	0.58	0.84	0.81	0.77	0.79	0.56	0.39
c4.large	0.40	0.37	0.37	0.46	0.55	0.51	0.50	0.58	0.53	0.65	0.63	0.59	0.81	0.79	0.75	0.78	0.56	0.40
t2.medium	0.43	0.40	0.40	0.49	0.56	0.52	0.53	0.59	0.55	0.66	0.62	0.59	0.74	0.70	0.68	0.69	0.48	0.31
m3.medium	0.31	0.30	0.30	0.34	0.42	0.39	0.36	0.43	0.40	0.50	0.47	0.44	0.57	0.55	0.51	0.86	0.77	0.57
t2.small	0.38	0.36	0.37	0.43	0.50	0.47	0.45	0.52	0.49	0.61	0.58	0.54	0.66	0.63	0.74	0.78	0.87	0.52
t2.micro	0.31	0.30	0.29	0.34	0.37	0.36	0.37	0.40	0.38	0.47	0.46	0.45	0.53	0.52	0.50	0.68	0.63	0.46



(a) AWS m3.large (2 core) (b) AWS m3.xlarge (4 core) (c) AWS m3.2xlarge (8 core)

(d) GCE n1-standard-2 (2 core) (e) GCE n1-standard-4 (4 core) (f) GCE n1-standard-8 (8 core)

Figure 7. Bare and *Docker* experiments and TCP standard receive window of 83780 bytes (dotted lines). *Weave* data is not plotted for readability reasons.

REFERENCES

[1] N. Kratzke, "About microservices, containers and their underestimated impact on network performance," CLOUD COMPUTING 2015, 2015, pp. 165–169.

[2] Docker. Last access 9th Nov. 2015. [Online]. Available: https://docker.com

[3] Weave. Last access 9th Nov. 2015. [Online]. Available: https://github.com/zettio/weave

[4] S. Newman, Building Microservices. O'Reilly and Associates, 2015.

[5] M. Fowler and J. Lewis. Microservices. Last access 17th Jul. 2015. [Online]. Available: http://martinfowler.com/articles/microservices.html [retrieved: March, 2014]

[6] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors," SIGOPS Oper. Syst. Rev., vol. 41, no. 3, Mar. 2007, pp. 275–287.

[7] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," IBM Research Division, Austin Research Laboratory, Tech. Rep., 2014.

[8] Mesos. Last access 10th Dec. 2015. [Online]. Available: https://mesos.apache.org

[9] Coreos. Last access 9th Nov. 2015. [Online]. Available: https://coreos.com

[10] Kubernetes. Last access 17th Jul. 2015. [Online]. Available: https://github.com/GoogleCloudPlatform/kubernetes

[11] N. Kratzke, "A lightweight virtualization cluster reference architecture derived from open source paas platforms," Open Journal of Mobile Computing and Cloud Computing (MCCC), vol. 1, no. 2, 2014, pp. 17–30.

[12] R. Peinl, F. Holzschuher, and F. Pfitzer, "Docker cluster management - state of the art and own solution," Journal of Grid Computing, 2015.

[13] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[14] N. Kratzke. Ping pong - a distributed http-based and rest-like ping-pong system for test and benchmarking purposes. Last access 9th Nov. 2015. [Online]. Available: https://github.com/nkratzke/pingpong

[15] ——, "Lightweight virtualization cluster - howto overcome cloud vendor lock-in," Journal of Computer and Communication (JCC), vol. 2, no. 12, oct 2014.

[16] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 295–308. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972488

[17] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," Communications Surveys Tutorials, IEEE, vol. 17, no. 1, Firstquarter 2015, pp. 27–51.

[18] Open vswitch. Last access 9th Nov. 2015. [Online]. Available: http://openvswitch.org

[19] Flannel. Last access 9th Nov. 2015. [Online]. Available: https://github.com/coreos/flannel

[20] Project calico. Last access 9th Nov. 2015. [Online]. Available: http://www.projectcalico.org/

[21] D. Mosberger and T. Jin, "Httperf - a tool for measuring web server performance," SIGMETRICS Perform. Eval. Rev., vol. 26, no. 3, Dec. 1998, pp. 31–37. [Online]. Available: http://doi.acm.org/10.1145/306225.306235

[22] G. Memik, W. H. Mangione-Smith, and W. Hu, "Netbench: A benchmarking suite for network processors," in Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design, ser. ICCAD '01. Piscataway, NJ, USA: IEEE Press, 2001, pp. 39–42.

[23] J. Verdú, J. Garcí, M. Nemirovsky, and M. Valero, "Architectural impact of stateful networking applications," in Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems, ser. ANCS '05. New York, NY, USA: ACM, 2005, pp. 11–18.

[24] K. Velásquez and E. Gamess, "A Comparative Analysis of Network Benchmarking Tools," Proceedings of the World Congress on Engineering and Computer Science 2009 Vol I, Oct. 2009. [Online]. Available: http://www.iaeng.org/publication/WCECS2009/WCECS2009_pp299-305.pdf

[25] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, Nov 2010, pp. 159–168.

[26] G. Wang and T. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in INFOCOM, 2010 Proceedings IEEE, March 2010, pp. 1–9.

[27] D. Jayasinghe, S. Malkowski, J. LI, Q. Wang, Z. Wang, and C. Pu, "Variations in performance and scalability: An experimental study in iaas clouds using multi-tier workloads," Services Computing, IEEE Transactions on, vol. 7, no. 2, April 2014, pp. 293–306.

[28] Berkley Lab, "iPerf - The network bandwidth measurement tool," https://iperf.fr, 2015.

[29] Sun Microsystems, "uperf - A network performance tool," http://www.uperf.org, 2012.

[30] netperf.org, "The Public Netperf Homepage," http://www.netperf.org, 2012.

[31] N. Kratzke and P.-C. Quint, "About automatic benchmarking of iaas cloud service providers for a world of container clusters," Journal of Cloud Computing Research, vol. 1, no. 1, 2015, pp. 16–34. [Online]. Available: http://jccr.uscip.us/PublishedIssues.aspx

[32] HP Labs, "httperf - A tool for measuring web server performance," http://www.hpl.hp.com/research/linux/httperf/, 2008.

[33] Apache Software Foundation, "ab - Apache HTTP server benchmarking tool," http://httpd.apache.org/docs/2.2/programs/ab.html, 2015.

[34] D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment," J. Supercomput., vol. 64, no. 3, Jun. 2013, pp. 835–848. [Online]. Available: http://dx.doi.org/10.1007/s11227-011-0625-1

[35] S. K. Garg, S. Versteeg, and R. Buyya, "Smicloud: A framework for comparing and ranking cloud services," in Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on. IEEE, 2011, pp. 210–218.

[36] C.-K. Ke, Z.-H. Lin, M.-Y. Wu, and S.-F. Chang, "An optimal selection approach for a multi-tenancy service based on a sla utility," in Computer, Consumer and Control (IS3C), 2012 International Symposium on. IEEE, 2012, pp. 410–413.

[37] A. Afshari, M. Mojahed, and R. M. Yusuff, "Simple additive weighting approach to personnel selection problem," International Journal of Innovation, Management and Technology, vol. 1, no. 5, 2010, pp. 511–515.

[38] C. Quinton, N. Haderer, R. Rouvoy, and L. Duchien, "Towards multi-cloud configurations using feature models and ontologies," in Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds. ACM, 2013, pp. 21–26.

[39] C.-W. Chang, P. Liu, and J.-J. Wu, "Probability-based cloud storage providers selection algorithms with maximum availability," in Parallel Processing (ICPP), 2012 41st International Conference on. IEEE, 2012, pp. 199–208.

[40] J. Siegel and J. Perdue, "Cloud services measures for global use: the service measurement index (smi)," in SRII Global Conference (SRII), 2012 Annual. IEEE, 2012, pp. 411–415.

[41] R. Karim, C. Ding, and A. Miri, "An end-to-end qos mapping approach for cloud service selection," in Services (SERVICES), 2013 IEEE Ninth World Congress on. IEEE, 2013, pp. 341–348.

[42] J. Yang, W. Lin, and W. Dou, "An adaptive service selection method for cross-cloud service composition," Concurrency and Computation: Practice and Experience, vol. 25, no. 18, 2013, pp. 2435–2454.

[43] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," Computers, IEEE Transactions on, vol. PP, no. 99, 2015, pp. 1–1.

[44] J. Stone, "Tachyon parallel / multiprocessor ray tracing system," http://jedi.ks.uiuc.edu/~johns/raytracer, 1995, accessed May 20, 2015.

[45] J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, Dec. 1995, pp. 19–25.

[46] W. D. Norcott and D. Capps, "Iozone filesystem benchmark," www.iozone.org, vol. 55, 2003.

[47] D. Lin, "An information-theoretic definition of similarity," in Proceedings of the Fifteenth International Conference on Machine Learning, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 296–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=645527.657297

[48] Dart. Last access 9th Nov. 2015. [Online]. Available: https://www.dartlang.org/

[49] Apachebench. Last access 9th Nov. 2015. [Online]. Available: http://httpd.apache.org/docs/2.2/programs/ab.html

[50] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org/

[51] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," RFC 1323 (Proposed Standard), Internet Engineering Task Force, May 1992, obsoleted by RFC 7323. [Online]. Available: http://www.ietf.org/rfc/rfc1323.txt

[52] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP Extensions for High Performance," RFC 7323 (Proposed Standard), Internet Engineering Task Force, Sep. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7323.txt

Distance Sensor Assistance to GPS Positioning

Yasuhiro Ikeda, Hiroyuki Hatano, Masahiro Fujii, Atsushi Ito, Yu Watanabe
Graduate School of Engineering,
Utsunomiya University
e-mail:{ikeda@degas., hatano@, fujii@, at.ito@, yu@}is.utsunomiya-u.ac.jp

Tomoya Kitani
Graduate School of Informatics,
Shizuoka University
e-mail:t-kitani@ieee.org

Toru Aoki
Research Institute of Electronics,
Shizuoka University
e-mail:rtaoki@rie.shizuoka.ac.jp

Hironobu Onishi
Graduate School of Engineering,
Shizuoka University
e-mail:onishi@hatanolab.eng.shizuoka.ac.jp

***Abstract*—Global Positioning System (GPS) requires the signals from at least four observable satellites with good condition. However, in urban areas, the number of the observable satellites in line-of-sight decreases because urban areas have many buildings. In such case, the position estimation cannot be performed well. In order to estimate even if the good observable satellites are decreased, we use distance sensors. Many vehicles have the distance sensors that can measure traveling distances. In our proposal, the distance sensors assist in calculating own position. By the combination of GPS and the distance sensors, the proposed method can continue robust positioning even if we are in urban areas. Our experimental results will evaluate our effectiveness of the proposal.**

Keywords*–*GPS; Positioning; Urban canyon; Lack of observable satellites.

I. Introduction

In recent years, location-based services have been increasing. These services require the information of user's position. Car navigation systems can be given as an example. In general, these services use the Global Positioning System (GPS). GPS can estimate user's position by using flying satellites around the Earth. We also focus on a novel GPS positioning in this paper [1].

The position estimation by GPS calculates the position, based on the measurement of the distances between the GPS receiver and the GPS satellites. In order to perform the position estimation, GPS receiver requires at least four satellites that can be received in line-of-sight [2]. However, in urban areas, the number of the observable satellites decreases because of buildings. Also, in urban canyon, GPS signals from the observable satellites tend to degrade because of multi-path propagation. The signals from satellites in non-line-of-sight should not be used because the errors are included in the propagated paths. In order to avoid degraded signals, the number of direct-path satellites that we prefer to use is more decreased. In such cases, the position estimation cannot be performed well. For example, the estimator cannot estimate the receiver's position if the number of observable satellites becomes less than three. Or, the performance becomes worse compared to the conditions in open sky where we can observe more than four satellites. Actually, conventional GPS systems rely on other information such as map information, base stations of assisted cellular networks, or Wi-Fi networks against the bad GPS measurement. In this paper, we will propose another assistance by a sensor data.

Many vehicles have wheel rotation sensors that can measure traveling distances. We proposed the novel positioning method that uses the information of the previous traveling distance [3][4][1]. Our proposed method uses the information of both the distance sensor and the previous position that is estimated under good condition. Higher accuracy in position estimation is expected by our proposed method, even when the number of observable good satellites is decreased. Also, it can be expected to prevent the impossibility of the position estimation when the number of the observable satellites becomes three. Moreover, better position estimation is expected by our proposed method, even when available satellites includes large error. By both the traveling distance and previous position, the proposal can add a quasi-satellite to GPS satellites. So, we can expect robust positioning by assistance of the sensor data.

In this paper, we will present the proposal's performance by field experiments. In the experiments, the position estimation is performed by traveling on a bicycle. From experimental results, we will evaluate the effectiveness of our proposed method.

This paper organized as follows. In Section II, we will introduce related works briefly. In Section III, we will show the case that the number of the observable satellites becomes three as the worst case. Under the worst case of three satellites, we will show our proposal in detail. In Section IV, we will evaluate our proposed method under three satellites. In Section V, we will show application of the proposal. We will apply our proposal to the case that more satellites exist. In Section VI, we will evaluate in case that adding the quasi-satellite by the proposal to the available all satellites. Finally, Section VII summarizes the paper.

II. Related Work

In this section, we will show the related methods to improve the accuracy of the position estimation. Here, we are introducing the traditional and basic technologies.

A. Differential GPS (DGPS)

The Differential GPS (DGPS) is the method for improving the position estimation accuracy [5]. The DGPS uses the GPS base station. The GPS base station transmits the information of the error amount in the GPS measurement to near GPS receivers. Measuring of the error information is performed accurately at the GPS base station.

Generally, the position estimation by GPS calculates the position by using the measurement of the distances between the GPS receiver and the GPS satellites. However, some errors are included in the distance measurement. The distance errors by clock difference, the ionospheric delay, and the troposphere delay can be given as examples. The estimation accuracy of user's position can be improved by correcting the error information that is generated at the GPS base station. However, the GPS estimation is used in various locations, such as urban areas, rural areas, sea, and mountains. In the urban areas, the DGPS cannot correct the propagation delay caused by the reflection at buildings. Therefore, in such case, the position estimation cannot be performed well.

B. Dead Reckoning (DR)

The Dead Reckoning (DR) is the method of performing position estimation by the information of the relative movement [6]. In other words, DR uses the information how much we traveled from the previous position. Since the DR does not require any infrastructures, the DR is not limited to any area.

In vehicles, various sensors exist to detect the direction. Usually, the angular velocity is detected by the angular velocity sensor. The angular velocity can be calculated by integrating the traveling direction [7]. Also, it is possible to detect the direction by using a fiber optic gyroscope. The fiber optic gyroscope is a device for determining the direction by measuring the time difference of the light when the angular velocity is added to the fiber optic.

By using vehicle speed pulses, it is possible to detect the traveling distance. The vehicle speed pulse is the signal that is generated according to the rotational speed of the drive shaft of the vehicle. We can measure the traveling distance based on the circumference of the tire and the vehicle speed pulse.

In the DR, the relative position can be estimated by using the moved direction and the traveled distance. The DR is often used with a map matching technique, as described in the next section. The combination of both DR and map matching are often used in the car navigation systems.

C. Map Matching

The map matching is the method for finding the appropriate position of the vehicle on the road by using the map information [8]. It is used in combination with the position estimation methods such as GPS and DR.

Currently, the map matching and the DR are commonly used in the car navigation systems [14]. The DR is a system for determining the relative position from the previous position. In DR, the error accumulates when the error occurs in the distance sensor and the direction sensor. This problem can be solved by using the map matching, but the map matching cannot be used in a place that does not have the map information. So, in order to estimate the absolute position, the DR with the map matching is often used with GPS. However, the sensor data from the DR does not improve the positioning accuracy of GPS directly.

As other works for multipath propagation in urban area, there are a lot of researches which can mitigate the multipath interference. For example, techniques which focus on signal tracking on a receiver or channel estimation by multiple correlators are shown[9], [10]. Moreover, the mitigation methods of Direction-of-Arrival (DoA) estimation by array antenna are also shown[11], [12]. The purpose of these methods is to prevent the degradation of the positioning performance in case of the multipath propagation with LOS satellites. Also, most of recent cellular phones use assisted-GPS (A-GPS), which is based on 3G/4G connection [13]. By using A-GPS, users can estimate own position rapidly under the environment of weak or bad satellite signals.

As a new method, we want to improve the position estimation accuracy by adding the sensor data, to the GPS position estimation directly. Here, we use a distance sensor. We proposed the positioning method that can estimate the absolute position by the combination of the distance sensor and GPS [3][4][1]. By our proposed method, the absolute position can be estimated even if only GPS cannot estimate own position exactly because of the bad environment. The bad environment for the conventional GPS is under lack of the observable satellites in line-of-sight. This case often happens in urban areas. The worst case is that the number of the observable satellites is three. The proposed method can estimate the own position but the conventional cannot estimate. We will treat this worst case in Sections III and IV. Of course, if the number of satellites is more than four, our proposed method may estimate well by assistance of the distance sensor. We will introduce this case in Sections V and VI. In our proposal method, we will keep estimating the user's absolute position by assistance of the distance sensor.

III. Proposed Method Under Three Satellites

In this section, first, we will show the problems of the position estimation by GPS. As a worst case, we assume that the number of the observable good satellites is three. Thereafter, we will show our proposed method that uses the distance sensor and the previous position information.

A. Problems of Position Estimation by GPS

The GPS positioning estimates the receiver's position based on the distances between the receiver and the satellites [15]. For estimating the 3D position of the receiver, the receiver needs three relations, that is, three satellites. Moreover, the receiver needs to estimate own clock error because the general receivers are equipped with inexpensive crystal clocks. Therefore, the GPS position

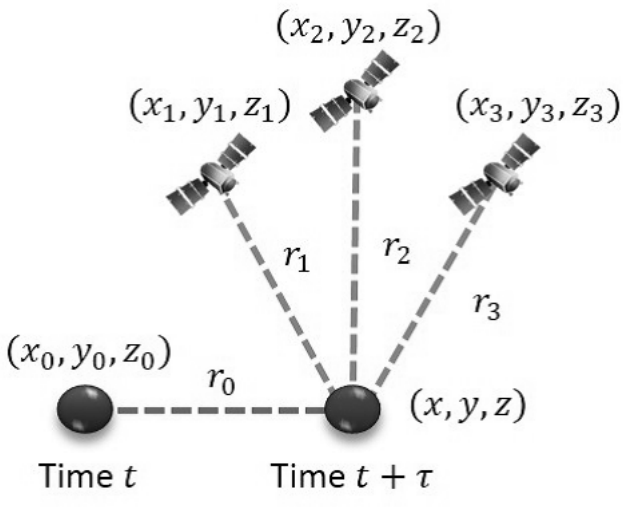


Figure 1. Assumed situation of our research.

estimation needs four relations, that is, four satellites. For carrying out the positioning calculation, the GPS position estimation needs at least four observable satellites. The observable satellite means the satellite that can receive its signal in line-of-sight. However, in urban areas, the number of the observable satellites are decreased because the receiver cannot observe the low elevation satellites due to the interference of buildings. Therefore, the receiver is not always possible to observe more than or equal to the four satellites in line-of-sight. So, the number of the observable satellites tends to decrease. Such decrement results in degradation of the position estimation accuracy. As a worst case, the receiver cannot estimate its own position if the number of the observable satellites becomes less than four. This is the most common problem in urban area.

To solve the above problem, we propose the novel GPS estimation that uses the distance sensor and the previous position information. The proposal method uses the previous receiver's position. We can measure the traveling distance from the previous position by the distance sensor. We assume the previous position as the quasi-satellite, which uses both the previous position and the traveled distance. By the proposed method, we expect that the position estimation is possible even if the number of the observable satellites are three. In addition, we also expect the improvement of position estimation accuracy when the number of the observable satellites are low. The detailed procedure is shown in the next subsection.

B. Proposed Position Estimation Algorithm

The proposed method is assumed to be used in position estimation on vehicles such as cars or bikes. The assumed situation of our research is shown in Figure 1. We consider 2 observation times (the time t and $t + \tau$). The position coordinate of the receiver at the time t is (x_0, y_0, z_0). And the position coordinate of the receiver at the time $t + \tau$ is (x, y, z). We want to estimate the position (x, y, z). We define that the position of the i-th satellite is (x_i, y_i, z_i). By using the orbital information of the satellites that is contained in the satellite signals, the positions of the satellites can be defined. Also, the variables $r_i (i = 1, 2, 3)$ are the distances between the receiver and the satellites. On the other hand, the variables r_0 is the distance between the time t position and the time $t + \tau$ position. Here, we assume that the position was correctly estimated by the adequate satellites at the time t. After traveling, we assume that the observable satellites are decreased at the time $t + \tau$. The number of the observable satellites at the time $t + \tau$ are assumed as three. In this case, the position estimation becomes impossible because of lack of observable satellites.

In this paper, for the purpose of simple explanation, we assume that the number of the observable satellites are four at the time t. The GPS positioning at the time t uses the satellite positions and the distances between the receiver and the satellites. The true distance ρ_i between the i-th satellite and the receiver can be expressed as follows.

$$\rho_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \tag{1}$$

The clock error may be included to the distance between the receiver and the satellite. Therefore, the distance r_i can be expressed by (2).

$$r_i = \rho_i + s \tag{2}$$

Here, the clock error is represented as the parameter s. The unit of clock error s is meter. This equation can be applied to all the observable satellites. As the number of the observable satellites are four at the time t, the following four equations can be obtained.

$$\begin{cases} r_1 = \rho_1 + s \\ r_2 = \rho_2 + s \\ r_3 = \rho_3 + s \\ r_4 = \rho_4 + s \end{cases} \tag{3}$$

By solving (3), it is possible to find out the position of the receiver (x_0, y_0, z_0) and the clock error s [16].

In case of the time $t + \tau$, the above method is not applicable for the position estimation because the number of the observable satellites are three. Therefore, we will estimate the position (x, y, z) by adding the quasi-satellite. That is, we use the previous position (x_0, y_0, z_0) and the distance between the current position and the previous position r_0. The distance r_0 can be represented by (4).

$$r_0 = \sqrt{(x_0 - x)^2 + (y_0 - y)^2 + (z_0 - z)^2} \tag{4}$$

At the time $t + \tau$, we can observe the three satellites. So, we can obtain the three equations (2). By using the three relations based on (2) and (4), we can derive the following equations.

$$\begin{cases} r_1 = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2} + s \\ r_2 = \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2} + s \\ r_3 = \sqrt{(x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2} + s \\ r_0 = \sqrt{(x_0 - x)^2 + (y_0 - y)^2 + (z_0 - z)^2} \end{cases} \tag{5}$$

It is possible to estimate the position (x, y, z) by (5).

The advantage of the proposed method is that we can increase the available satellites by adding (4). In addition,

the proposed method only uses the GPS receiver and the distance sensor, no other infrastructure is needed. In our proposal, we need the distance between the current position and the previous position. In recent vehicles, it is easy to measure the traveling distance because most of the vehicles have distance sensors, such as speed pulses. Also, our proposal use the previous position. So, we note the proposal may accumulate the positioning errors that are based on the estimation errors of the previous position.

C. Calculation Process of Position Estimation

Because the simultaneous equation (5) is nonlinear, the solution can be obtained by the sequential approximation that is performed the linearization around the initial value. Here, the procedure of the sequential approximation is shown below. As a notation, subscripts of the right shoulder of the following variables indicate the times of the sequential approximation.

1) we prepare the suitable initial values x^0, y^0, z^0, s^0 about x, y, z, s.
2) By using the receiver position x^0, y^0, z^0 and the clock error s^0, we calculate the distances between the receiver and the satellites.

$$\begin{cases} r_1^0 = \sqrt{(x_1 - x^0)^2 + (y_1 - y^0)^2 + (z_1 - z^0)^2} + s^0 \\ r_2^0 = \sqrt{(x_2 - x^0)^2 + (y_2 - y^0)^2 + (z_2 - z^0)^2} + s^0 \\ r_3^0 = \sqrt{(x_3 - x^0)^2 + (y_3 - y^0)^2 + (z_3 - z^0)^2} + s^0 \\ r_0^0 = \sqrt{(x_0 - x^0)^2 + (y_0 - y^0)^2 + (z_0 - z^0)^2} \end{cases} \quad (6)$$

3) The residual error $\Delta r_i = r_i - r_i^0$ can be determined by using the distance $r_i (i = 0, 1, 2, 3)$, which is actually measured.
4) Since it is possible to approach the correct solution by compensating same amount corresponding to the residual error for x^0, y^0, z^0, s^0, the compensation amount is determined using the partial derivative about x, y, z, s.

$$\begin{array}{rcl} \frac{\partial r_i}{\partial x} & = & -\frac{(x_i - x)}{r_i} \\ \frac{\partial r_i}{\partial y} & = & -\frac{(y_i - y)}{r_i} \\ \frac{\partial r_i}{\partial z} & = & -\frac{(z_i - z)}{r_i} \\ \frac{\partial r_i}{\partial s} & = & \begin{cases} 1 (i = 1, 2, 3) \\ 0 (i = 0) \end{cases} \end{array} \quad (7)$$

From (7), the compensation amount $\Delta x, \Delta y, \Delta z, \Delta s$ to update x^0, y^0, z^0, s^0 can be represented as follows.

$$\begin{cases} \Delta r_1 = \frac{\partial r_1}{\partial x}\Delta x + \frac{\partial r_1}{\partial y}\Delta y + \frac{\partial r_1}{\partial z}\Delta z + \frac{\partial r_1}{\partial s}\Delta s \\ \Delta r_2 = \frac{\partial r_2}{\partial x}\Delta x + \frac{\partial r_2}{\partial y}\Delta y + \frac{\partial r_2}{\partial z}\Delta z + \frac{\partial r_2}{\partial s}\Delta s \\ \Delta r_3 = \frac{\partial r_3}{\partial x}\Delta x + \frac{\partial r_3}{\partial y}\Delta y + \frac{\partial r_3}{\partial z}\Delta z + \frac{\partial r_3}{\partial s}\Delta s \\ \Delta r_0 = \frac{\partial r_0}{\partial x}\Delta x + \frac{\partial r_0}{\partial y}\Delta y + \frac{\partial r_0}{\partial z}\Delta z \end{cases} \quad (8)$$

Here, the simultaneous equation (8) can be represented by the matrix form in order to simplify handling. We define the vectors $\Delta\vec{x} = [\Delta x, \Delta y, \Delta z, \Delta s]^{\mathrm{T}}$ and $\Delta\vec{r} = [\Delta r_1, \Delta r_2, \Delta r_3, \Delta r_0]^{\mathrm{T}}$ (the notation T expresses a transpose), the equation (8) can be expressed as follow.

$$G\Delta\vec{x} = \Delta\vec{r} \quad (9)$$

Here, the matrix G is usually called as the observation matrix or the design matrix. The matrix G can be expressed as follows.

$$\begin{aligned} G &= \begin{bmatrix} \frac{\partial r_1}{\partial x} & \frac{\partial r_1}{\partial y} & \frac{\partial r_1}{\partial z} & \frac{\partial r_1}{\partial s} \\ \frac{\partial r_2}{\partial x} & \frac{\partial r_2}{\partial y} & \frac{\partial r_2}{\partial z} & \frac{\partial r_2}{\partial s} \\ \frac{\partial r_3}{\partial x} & \frac{\partial r_3}{\partial y} & \frac{\partial r_3}{\partial z} & \frac{\partial r_3}{\partial s} \\ \frac{\partial r_0}{\partial x} & \frac{\partial r_0}{\partial y} & \frac{\partial r_0}{\partial z} & \frac{\partial r_0}{\partial s} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{(x_1 - x)}{r_1} & -\frac{(y_1 - y)}{r_1} & -\frac{(z_1 - z)}{r_1} & 1 \\ -\frac{(x_2 - x)}{r_2} & -\frac{(y_2 - y)}{r_2} & -\frac{(z_2 - z)}{r_2} & 1 \\ -\frac{(x_3 - x)}{r_3} & -\frac{(y_3 - y)}{r_3} & -\frac{(z_3 - z)}{r_3} & 1 \\ -\frac{(x_0 - x)}{r_0} & -\frac{(y_0 - y)}{r_0} & -\frac{(z_0 - z)}{r_0} & 0 \end{bmatrix} \end{aligned} \quad (10)$$

The compensation amount $\Delta x, \Delta y, \Delta z, \Delta s$ in (8) can be derived by multiplying the inverse matrix of G from the left of (9). Therefore, the compensation amount $\Delta x, \Delta y, \Delta z, \Delta s$ can be determined by solving (11).

$$\Delta\vec{x} = G^{-1}\Delta\vec{r} \quad (11)$$

5) The initial values x^0, y^0, z^0, s^0 are updated by $\Delta x, \Delta y, \Delta z, \Delta s$ as follows.

$$\begin{array}{rcl} x^1 & = & x^0 + \Delta x \\ y^1 & = & y^0 + \Delta y \\ z^1 & = & z^0 + \Delta z \\ s^1 & = & s^0 + \Delta s \end{array} \quad (12)$$

6) After updating the initial value to x^1, y^1, z^1, s^1, we return to the Procedure 2. These procedures are repeated until $\Delta x, \Delta y, \Delta z, \Delta s$ becomes enough small.

By following the above procedure, our proposed method has the possibility to calculate the receiver's position (x, y, z). In our experience, the solution can converge by repeating several times even if the initial values are $x^0 = y^0 = z^0 = s^0 = 0$.

IV. Characterization by Field Experiment I

A. Setup and Environment

The experiments were conducted in order to evaluate the ability and the effectiveness of the proposed method. In the proposed method, the position estimation is performed while updating the traveling distance and the previous position. The assumed environment is an urban area. There are many buildings in urban areas. The satellites with low elevations tend to be shaded by the buildings. As the worst case, the number of the observable satellites is only three. We use three satellites with high elevations.

In the experiment, we are using a bicycle as the moving vehicle. The bicycle is shown in Figure 2. As shown in Figure 2, a GPS antenna is attached to the loading platform. Also, the cycle computer is attached to the front wheel. The cycle computer is shown in Figure 3. The cycle computer is a device that senses a magnet mounted on the spokes of the tire to generate a pulse after each rotation. The bicycle also has a data logger system. Each generated pulse is saved in the data logger. The sampling interval of the data logger is 1 ms. The example of the saved pulses is shown in Figure 4. From Figure 4, the cycle computer outputs 0V when the magnet passes the front of the sensor. Except above, the cycle computer usually outputs 2V. We

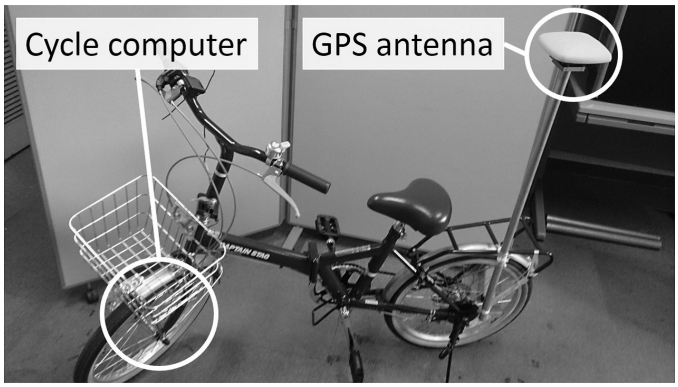


Figure 2. Experimental vehicle.

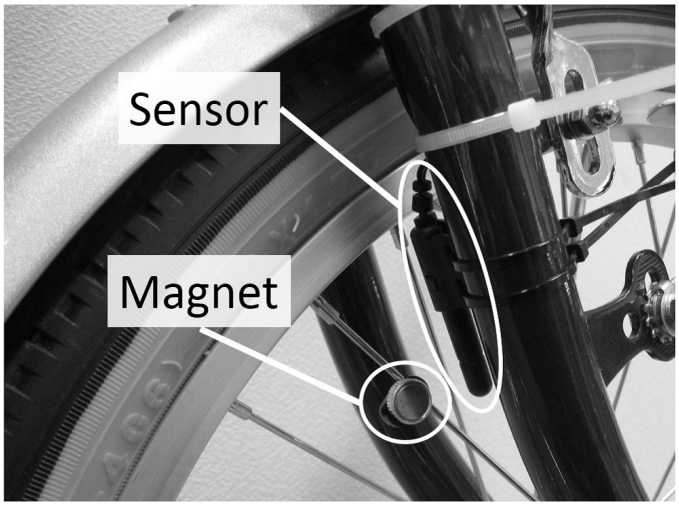


Figure 3. Cycle computer.

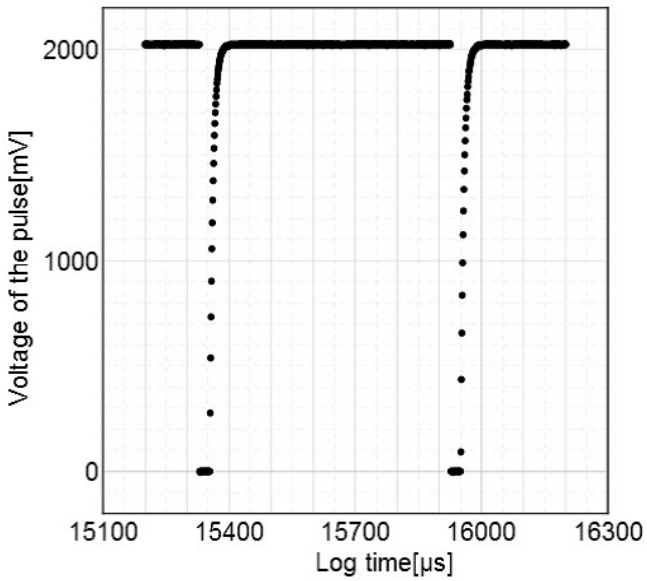


Figure 4. Waveform from the cycle computer.

TABLE I. SPECIFICATIONS OF THE EXPERIMENT ENVIRONMENT

GPS receiver	JAVAD GNSS DELTA-G3T
Number of observable satellites	8 satellites
Total traveling time	50 s
Estimation interval of GPS receiver	1 s
Data logger	EasySync DS1M12
Cycle Computer	CATEYE CC-VL820 VELO 9
Sampling interval of the data logger	1 ms
Circumference of the tire of the bicycle	1.515m

can recognize the rotation of the tire by the pulses. Based on these pulses, the distance r_0 can be measured. The duration between an edge of a pulse and that of the next pulse is equal to the circumference of the tire. The traveling distance r_0 is calculated for each position estimation by using the circumference of the tire.

The experiment has been conducted under an open sky. The distance r_0 had measured while traveling by the bicycle. The position estimation of the proposed method is performed using the three satellites with high elevation and the previous position. For comparison, the positions are also estimated by the conventional method with all the observable satellites. As mentioned in Section II, there are the past researches. However, in this paper, we compare the stand-alone GPS estimation as standard.

A total distance of the experimental riding is 100 meters. In the first 20 meters, we rode toward east straightly. Then, we turned right. In the next 80 meters, we rode toward south straightly again. The cycle computer has a function that displays the speed according to the rotational speed of the tire. We kept the speed of the bicycle 10 km/h. From the start to the end, the time is 50 seconds. Table I summarizes the parameters of the experiment environment.

By using the data obtained in the experiment, the position estimation is performed per second. The GPS receiver can output the distance between the receiver and the satellites. The output distance includes some errors, such as the ionospheric delay error and the tropospheric delay error (so, the output distance is often called as the pseudo range). The ionospheric delay can be estimated by the transmitted messages from the satellites because the messages have coefficients of equations, which are modeled as the ionospheric delay. So, in this paper, we subtract the modeled ionospheric delay from the output distance. Similarly, we subtract the modeled tropospheric delay from the output distance. We use the remaining distance as the distance r_i For comparison, the position estimation using all satellites was also calculated per second. In the proposed method, the position coordinates of the start is estimated by using the four satellites with high elevation.

B. Position Estimation Results

Figure 5 shows the results of the proposed and conventional method of position estimation. The origin of Figure 5 is the starting point. The positioning results are plotted every second. According to Figure 5, by using the proposed

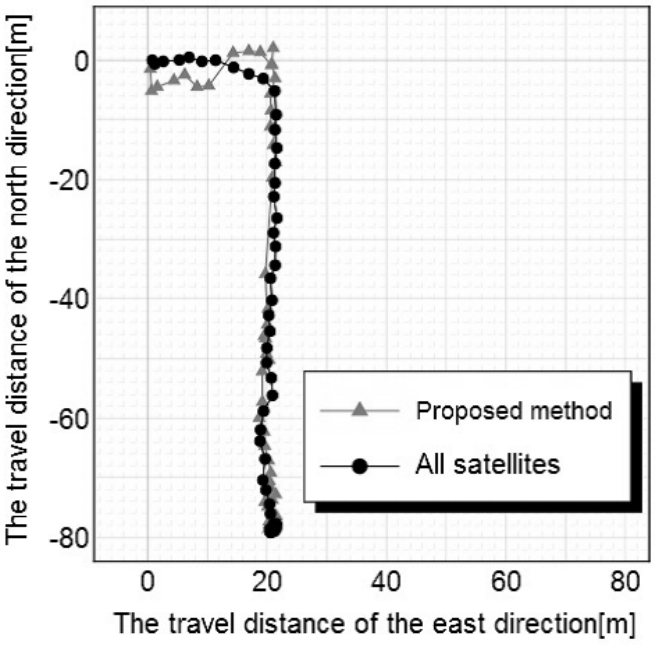


Figure 5. Position estimation results of all satellites and the proposed method.

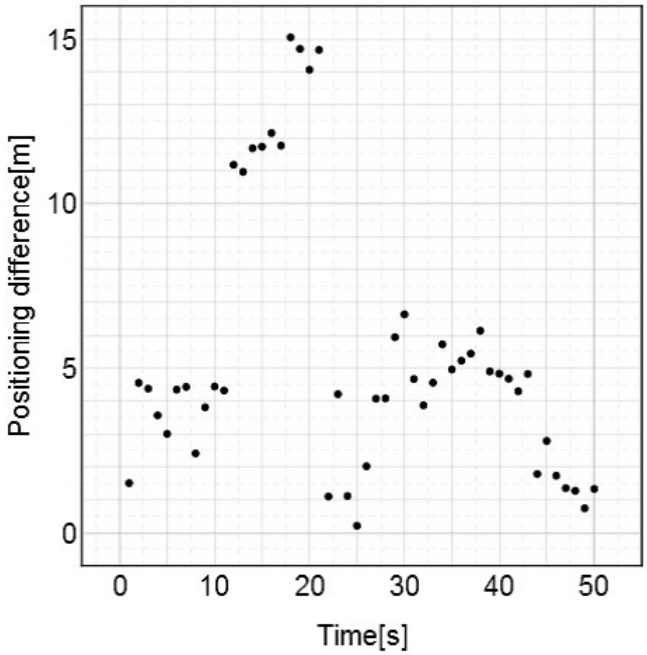


Figure 6. Positioning difference.

method, the position estimation can be performed even when the number of the observable satellites are three. First, the bicycle is moving towards east, from the start point. After going 20 meters straightly, the direction is changed to the south. Finally, the vehicle stops when the traveling distance becomes 80 meters from the turned point. As we can see, the proposal method can keep estimating the position when the direction of the traveling is changed while traveling.

In Figure 6, the positioning difference between the estimated position by the proposed method and the position by all the satellites is shown. The positioning difference is defined as Euclidean distance between both the positions. Figure 6 is plotted every second. The total traveling time is 50 seconds. The positioning difference is 5 meters or less until 11 seconds from the start time. In addition, from 22 seconds to 50 seconds, the positioning difference also under 5 meters. Also, a few samples are 7 meters or less. From 12 seconds to 21 seconds, the positioning difference is over 10 meters.

As another viewpoint of discussion, Figure 7 shows the cumulative probability distribution in order to check the distribution of the positioning difference. From Figure 7, 76 percent of the positioning differences are 5 meters or less. The rate of the positioning differences over 10 meters is about 17 percent.

By considering these results, the proposed method is able to estimate the receiver's position by using the previous position and the three satellites with high elevation. However, there are some cases when the positioning differences are more than 10 meters. These large differences are a problem that will have to be resolved. One of the above reasons is the satellites constellation. Generally, in case of the four satellites, the good satellites constellation

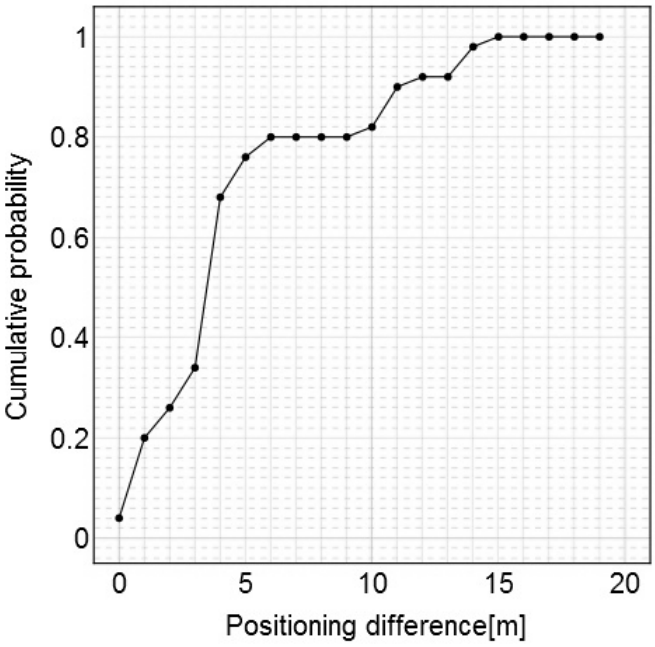


Figure 7. Cumulative probability distribution of the positioning difference.

can be presented as follows [17].

- One satellite is near zenith, that is, with high elevation.
- Other three satellites are distributed and surrounded uniformly with low elevations.

In this experiment, three satellites with high elevation have

been selected. A better selection has to be consider for the above appropriate satellite constellation. By a better selection, we expect that the proposal can become better.

We now investigate other reasons why the large differences occur. In this paper, the properties were evaluated by comparing the position estimation results of using all satellites. However, the measurement error of the estimation results by all satellites may be included. In oder to investigate in more detail, it is necessary to evaluate the characteristics by comparing the true position with the proposed method.

The proposed process is not much different from the conventional process. In the proposed method, we just use the traveling distance instead of the range from a satellites. So, there is no big difference in calculation time compared to the conventional positioning. We hope that our proposal can be calculated in real-time.

V. Application of proposal

The proposed method in Section III was used for three satellites. In this section, we will apply the assistance of the distance sensor by the proposal for estimating own position by available all satellites. By using both the distance sensor's data and the previous position, we can add the quasi-satellite to the available satellites. That is, we can increase the number of available satellites. For example, we can estimate own position by 11 satellites (10 satellites and a quasi-satellite) even when we can observe 10 satellites.

The calculation equations are as follows. These are similar to (8).

$$\begin{cases} \Delta r_1 = \frac{\partial r_1}{\partial x}\Delta x + \frac{\partial r_1}{\partial y}\Delta y + \frac{\partial r_1}{\partial z}\Delta z + \frac{\partial r_1}{\partial s}\Delta s \\ \Delta r_2 = \frac{\partial r_2}{\partial x}\Delta x + \frac{\partial r_2}{\partial y}\Delta y + \frac{\partial r_2}{\partial z}\Delta z + \frac{\partial r_2}{\partial s}\Delta s \\ \Delta r_3 = \frac{\partial r_3}{\partial x}\Delta x + \frac{\partial r_3}{\partial y}\Delta y + \frac{\partial r_3}{\partial z}\Delta z + \frac{\partial r_3}{\partial s}\Delta s \\ \vdots \\ \Delta r_N = \frac{\partial r_N}{\partial x}\Delta x + \frac{\partial r_N}{\partial y}\Delta y + \frac{\partial r_N}{\partial z}\Delta z + \frac{\partial r_N}{\partial s}\Delta s \\ \Delta r_0 = \frac{\partial r_0}{\partial x}\Delta x + \frac{\partial r_0}{\partial y}\Delta y + \frac{\partial r_0}{\partial z}\Delta z \end{cases} \tag{13}$$

The variable $r_n, (n = 1, 2, \cdots, N)$ means the distances between the receiver and the satellites. The variable N is the number of the used satellites.

As mentioned before, the distance r_n is often called as the pseudo range because the distance r_n includes errors. The lower the elevation angle of the satellite is, the larger these errors become. Generally, the relation between the standard deviation of the errors in the pseudo range and the elevation angle can be approximated as:

$$\sigma(\theta) = \frac{0.8}{\sin\theta}, \tag{14}$$

where the variable θ means the elevation angle of the satellite [18]. In the positioning processes, we have to consider the bad influence by the above error. So, the calculation equations are expanded into the weighted equations as follows [18]:

$$\begin{cases} \frac{1}{\sigma_1}\Delta r_1 = \frac{1}{\sigma_1}\frac{\partial r_1}{\partial x}\Delta x + \frac{1}{\sigma_1}\frac{\partial r_1}{\partial y}\Delta y + \frac{1}{\sigma_1}\frac{\partial r_1}{\partial z}\Delta z + \frac{1}{\sigma_1}\frac{\partial r_1}{\partial s}\Delta s \\ \frac{1}{\sigma_2}\Delta r_2 = \frac{1}{\sigma_2}\frac{\partial r_2}{\partial x}\Delta x + \frac{1}{\sigma_2}\frac{\partial r_2}{\partial y}\Delta y + \frac{1}{\sigma_2}\frac{\partial r_2}{\partial z}\Delta z + \frac{1}{\sigma_2}\frac{\partial r_2}{\partial s}\Delta s \\ \frac{1}{\sigma_3}\Delta r_3 = \frac{1}{\sigma_3}\frac{\partial r_3}{\partial x}\Delta x + \frac{1}{\sigma_3}\frac{\partial r_3}{\partial y}\Delta y + \frac{1}{\sigma_3}\frac{\partial r_3}{\partial z}\Delta z + \frac{1}{\sigma_3}\frac{\partial r_3}{\partial s}\Delta s \\ \vdots \\ \frac{1}{\sigma_N}\Delta r_N = \frac{1}{\sigma_N}\frac{\partial r_N}{\partial x}\Delta x + \frac{1}{\sigma_N}\frac{\partial r_N}{\partial y}\Delta y + \frac{1}{\sigma_N}\frac{\partial r_N}{\partial z}\Delta z + \frac{1}{\sigma_N}\frac{\partial r_N}{\partial s}\Delta s \\ \frac{1}{\sigma_0}\Delta r_0 = \frac{1}{\sigma_0}\frac{\partial r_0}{\partial x}\Delta x + \frac{1}{\sigma_0}\frac{\partial r_0}{\partial y}\Delta y + \frac{1}{\sigma_0}\frac{\partial r_0}{\partial z}\Delta z \end{cases} \tag{15}$$

In order to calculate the compensation amount $\Delta\vec{x} = [\Delta x, \Delta y, \Delta z, \Delta s]^{\mathrm{T}}$, we used (11). In this section, we can determine the compensation amount as follows:

$$\Delta\vec{x} = \left(G^T W G\right)^{-1} G^T W \Delta\vec{r}, \tag{16}$$

where

$$W = \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 & 0 \\ \vdots & & \ddots & & \\ 0 & 0 & \cdots & \frac{1}{\sigma_N} & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{\sigma_0} \end{pmatrix}. \tag{17}$$

By using the above compensation, we can update the values x, y, z, s such as (12).

VI. Characterization by Field Experiment II

A. Setup and Environment

In order to confirm and evaluate the characteristics of the extended method in Section V, we conduct other simulations in addition to Section IV. In Section IV, we simulated the case that the number of satellites are three. In this section, we will apply our proposal to the estimation under available all satellites. By using the distance sensor's data, we add the quasi-satellite to the available satellites.

In urban canyon, the following problems are famous.

1) There are many tall buildings. In case of satellites with high elevations, a receiver can receive direct path signals from the satellites in line-of-sight. However, in case of satellites with middle elevations, the signal tends to be shielded by the buildings. So, the receiver cannot receive direct-path-signal. In case of indirect-path signal, estimated positions have large errors.
2) Also, the signals from satellites with low elevations tend to be shielded by the buildings. So, the propagated path includes large errors. Then the estimated positions have large errors.

By using our proposal, we can prepare the quasi-satellite. So, we can use the quasi-satellites instead of the above bad satellites.

The following simulations use the real satellites information that is measured in Section VI. The satellites constellaton, which is recorded in the measurement, is shown in Figure 8. The number of available satellites are eight. In the simulations, the weight of the quasi-satellite by the wheel sensor is set as 1. In other words, the weight W in (17) is set as 1.

In this section, we simulated the following two scenarios.

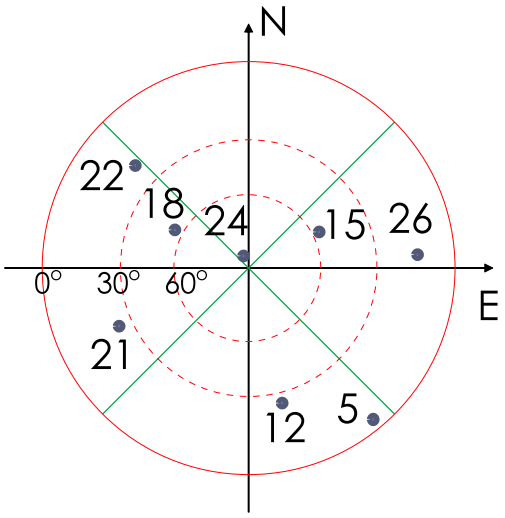


Figure 8. Satellites constellation.

Case A The satellite number PRN:18 includes large error. The elevation of the satellite PRN:18 is 58°, that is middle elevation. In order to simulate including the error, we add +10 meters to the measured range intentionally. In the proposal, the position can be estimated by using the quasi-satellite instead of PRN:18.

Case B The satellite number PRN:5 includes large error. The elevation of the satellite PRN:5 is 7°, that is very low elevation. In order to simulate including the error, we also add +10 meters to the measured range intentionally. In the proposal, the position can be estimated by using the quasi-satellite instead of PRN:5.

In both Case A and Case B, we treat the positioning result by all 8 satellites as true positions because the measurement is conducted under open-sky.

B. Position Estimation Results (Case A)

In Case A, the simulated results are shown in Figures 9 - 12. Figure 9 shows the plots that are estimated positions by both all 8 satellites and 8 satellites (10 meters is added to the range of PRN:18). Figure 11 shows the plots that are estimated positions by both all 8 satellites and 7 satellites (proposal without PRN:18). Figures 10 and 12 show the positioning differences to all 8 satellites.

Figure 9 shows the plots that are estimated positions by both all 8 satellites and 8 satellites (10 meters is added to the range of PRN:18). The origin is the start point. The positioning results are plotted every second. In results of the all 8 satellites, the bicycle is moving towards east, from the origin. After going 20 meters straightly from the start point, the direction is changed to the south. The bicycle stops when the traveling distance becomes 80 meters from the turned point. In results of the 8 satellites (10 meters is added to the range of PRN:18), the start point is 5 meters away from the origin. The trajectory is almost similar to that of the all 8 satellites case. The bicycle stop point of

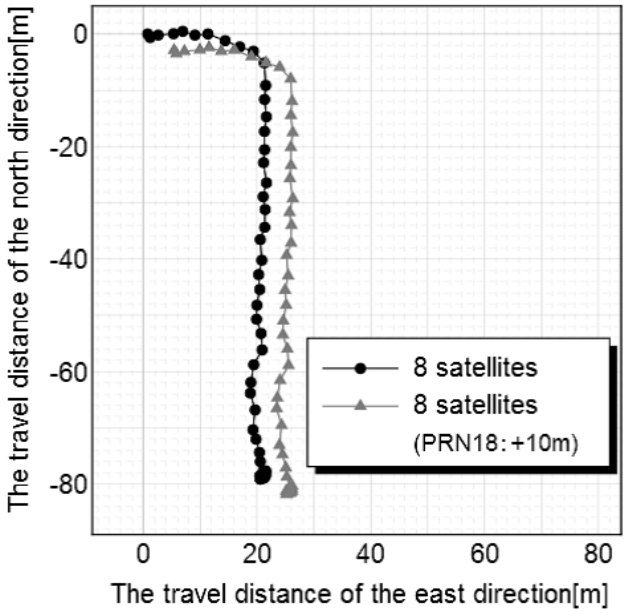


Figure 9. Positioning results (8 satellites vs 8 satellites(PRN 18: Indirect Path)).

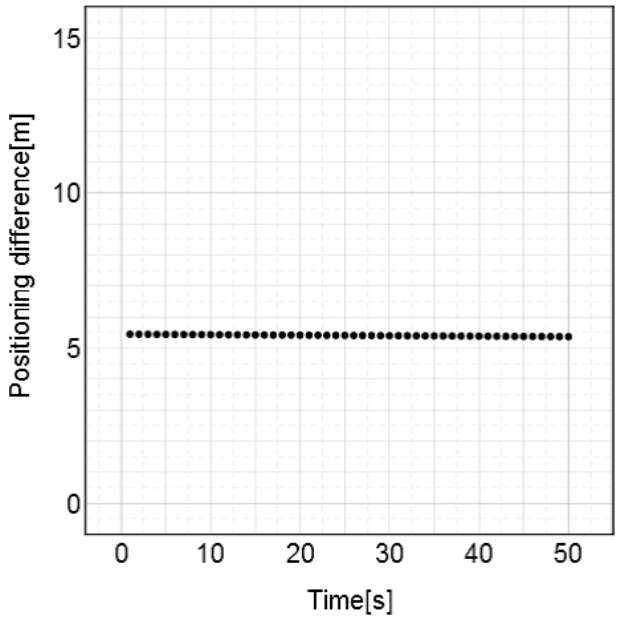


Figure 10. Positioning differences (8 satellites vs 8 satellites(PRN 18: Indirect Path)).

8 satellites (10 meters is added to the range of PRN:18) is 5 meters away from the stop point of all 8 satellites.

Figure 10 shows the positioning differences between the estimated position by the all 8 satellites and 8 satellites (10 meters is added to the range of PRN:18). Figure 10 is plotted every second. The total traveling time is 50 seconds. The positioning differences are around 6 meters or less from the start point to the stop point. The plots

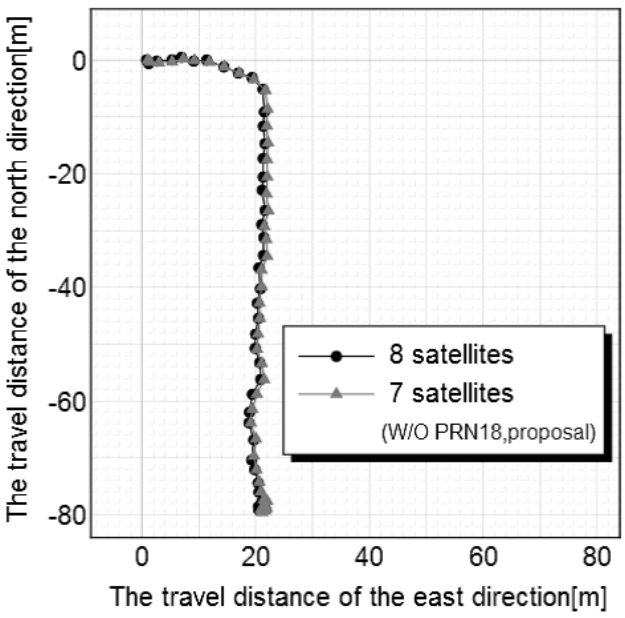


Figure 11. Positioning results (8 satellites vs 7 satellites(w/o PRN 18, proposal)).

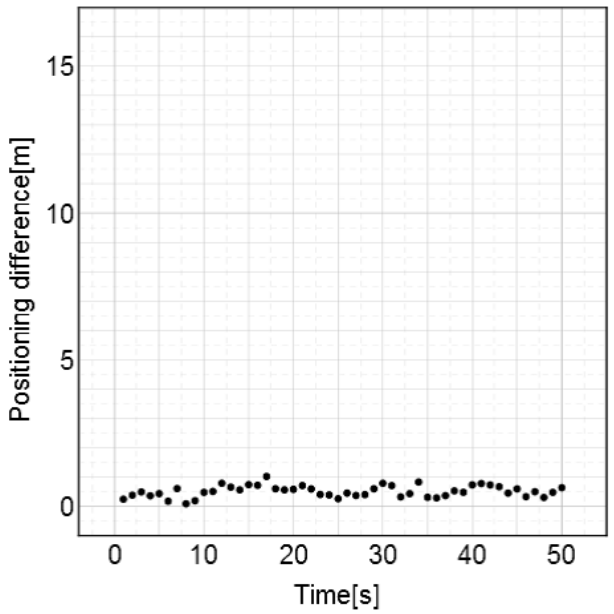


Figure 12. Positioning differences (8 satellites vs 8 satellites(w/o PRN 18, proposal)).

do not have large oscillation. The reason may be that the added error is a constant value. Average of the positioning difference is 5.40 meters. The positioning difference result must be influence by adding +10 meters.

Figure 11 shows the plots that are estimated positions by both all 8 satellites and 7 satellites (proposal without PRN:18). The origin is the start point. The positioning results are plotted every second. The result of the all 8 satellites case is same to that of all 8 satellites case in Figure 9. And, the results of the 7 satellites (proposal without PRN:18) are the result like the all 8 satellites.

In Figure 12, the positioning differences between the estimated position by the all 8 satellites and 7 satellites (proposal without PRN:18) are shown. Figure 12 is plotted every second and the total traveling time is 50 seconds. The positioning difference is around 1 meters or less from the starts to stop point. Average of the positioning difference is 0.51 meters.

By considering these results, the proposed method is effective in case A. Because, average of positioning difference in Figure 12 is less than that in Figure 10. The results suggest that the assistance of the distance sensor is effective instead of use of the satellite that includes large error. That is, the proposed method is effective.

C. Position Estimation Results (Case B)

In Case B, the simulated results are shown in Figures 13 - 16. Figure 13 shows the plots that are estimated positions by both all 8 satellites and 8 satellites (10 meters is added to the range of PRN:5). Figure 15 shows the plots that are estimated positions by both all 8 satellites and 7 satellites (proposal without PRN:5). Figures 14 and 16 show the positioning differences to all 8 satellites in case of the above simulations respectively.

Figure 13 shows the plots that are estimated positions by both all 8 satellites and 8 satellites (10 meters is added to the range of PRN:5). The origin is the start point. The positioning results are plotted every second. The moving of the bicycle is same in case of Section IV-B.

Figure 14 shows the positioning differences between the estimated position by the all 8 satellites and 8 satellites (10 meters is added to the range of PRN:5). Figure 14 is plotted every second. The total traveling time is 50 seconds. The positioning differences are almost constant because the added error is constant. Average of the positioning difference is about 0.26 meters.

Figure 15 shows the plots that are estimated positions by both all 8 satellites and 7 satellites (proposal without PRN:5). The plots in case of all 8 satellites is same to that in Figures 9, 11, and 13. The positioning results are plotted every second. The results of the 7 satellites (proposal without PRN:5) are close to the results of the all 8 satellites.

In Figure 16, the positioning differences between the estimated position by the all 8 satellites and 7 satellites (proposal without PRN:5) are shown. Figure 16 is plotted every second. Average of the positioning differences are 0.29 meters.

Comparing Figures 14 and 16, the rate in case that the positioning differences of the proposal are lower than that of the conventional with PRN:8, which includes the ranging error is 48%. If the satellites that include errors are enough low angle, the calculation process decides that the weight in (17) is extremely low. So, without our proposal, the influence of the satellite with low angle is not big problem. That is, the effective of the proposal is low. We can find that the proposal is always not effective.

VII. Conclusion and Future Work

In this paper, we proposed the novel positioning algorithm that used not only the GPS satellites but also the

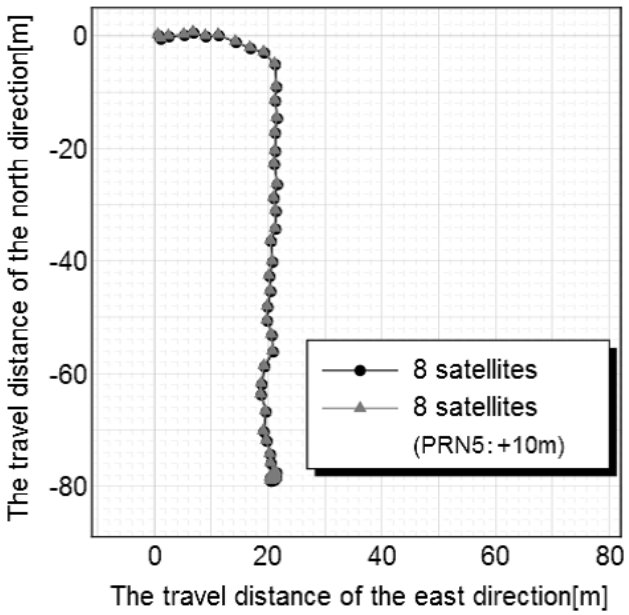


Figure 13. Positioning results (8 satellites vs 7 satellites(PRN 5: Indirect Path)).

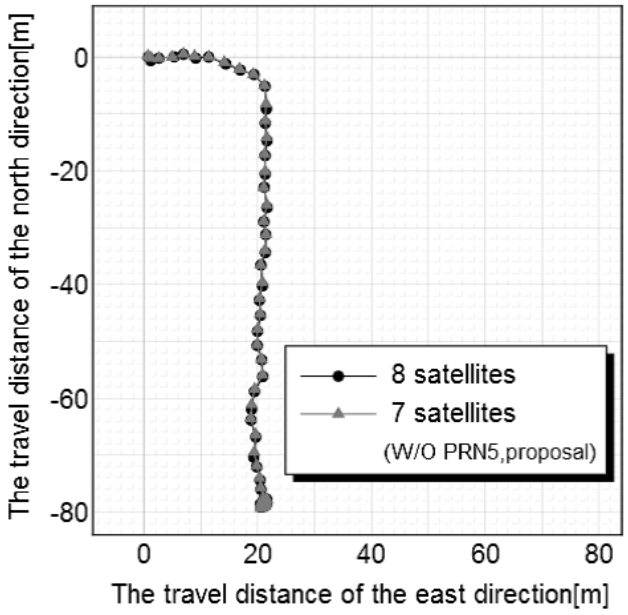


Figure 15. Positioning results (8 satellites vs 7 satellites(w/o PRN 5, proposal)).

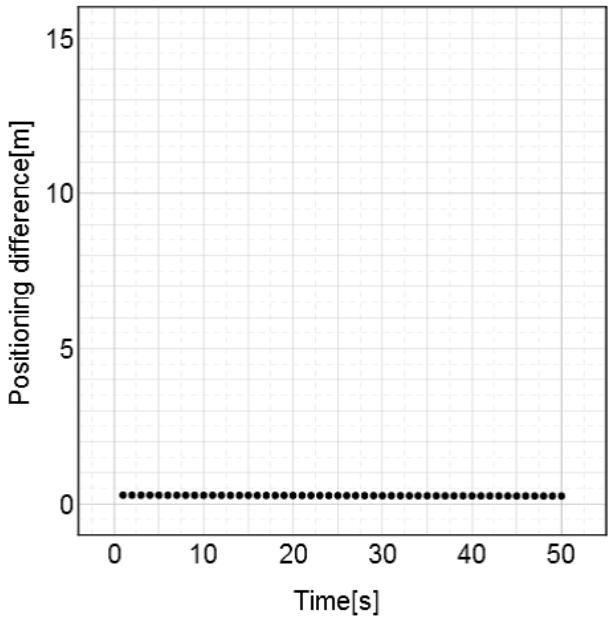


Figure 14. Positioning differences (8 satellites vs 8 satellites(PRN 5: Indirect Path)).

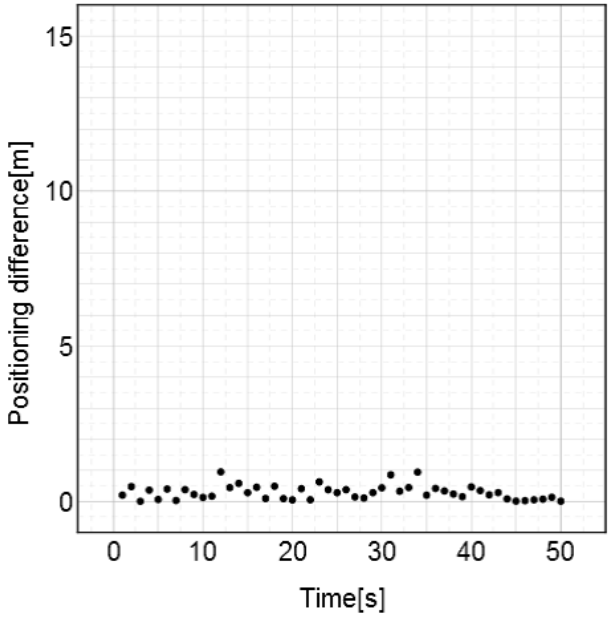


Figure 16. Positioning differences (8 satellites vs 8 satellites(w/o PRN 5, proposal)).

distance sensor information. In other words, the distance sensor assisted the GPS calculation process. By using our proposal, the problem that degrades GPS signals in urban canyon is improved. Because the proposal used the previous own position and the traveled distance that was measured by the distance sensor, the proposal can regard the sensor information as the quasi-satellite. The typical problem in urban area is the decrement of the observable satellites. The proposal can add the quasi-satellite. So, we can keep estimating own position even when the number of the observable satellites becomes low. Also, in case there are adequate number of the observable satellites, the quasi-satellite can help the conventional positioning. As a future work, we will try effective usage of the quasi-satellite. For example, we will find effective combination of the real satellites and quasi-satellite.

In order to evaluate the proposed method, we evaluated

two cases of both insufficient number of satellites and adequate number of satellites. In the insufficient case, we confirmed that the proposal was able to keep estimating own position robustly. In the adequate case, we confirmed that the quasi-satellite instead of the bad satellite that includes errors can reduce the positioning errors but the proposal was always not effective.

Acknowledgment

A part of this research is supported by the Cooperative Research Project of Research Institute of Electronics, Shizuoka University and the research promotion of the Murata science foundation.

References

[1] Y. Ikeda et al., "A Study on GPS Positioning Method with Assistance of a Distance Sensor," IARIA International Conference on Networks, 2015, pp. 109-114.

[2] T. H. Dixon, "An introduction to the global positioning system and some geological applications," Reviews of Geophysics, vol. 29, no. 2, 1991, pp. 249-276.

[3] H. Hatano, T. Kitani, M. Fujii, Y. Watanabe, and H. Onishi, "A Helpful Positioning Method with Two Gnss Satellites in Urban Area," IARIA International conference on Mobilie Services, Resources, and Users, 2013, pp. 41-46.

[4] H. Hatano et al., "Positioning Method by Two Gnss Satellites and Distance Sensor in Urban Area," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E98-A, no. 1, 2015, pp. 275-283.

[5] S. Skone and S. M. Shrestha, "Limitations in DGPS positioning accuracies at low latitudes during solar maximum," Geophysical Research Letters, vol. 29, no. 10, pp. 81-1-81-4, 2002, doi:10.1029/2001GL013854.

[6] Y. Fuke and E. Krotkov, "Dead Reckoning for a Lunar Rover on Uneven Terrain," IEEE, vol. 1, 1996, pp. 411-416.

[7] H. Kanoh, "Dynamic route planning for car navigation systems using virus genetic algorithms," IOS Press, vol. 11, no. 1, 2007, pp. 65-78.

[8] S. Kim and J. H. Kim, "Adaptive Fuzzy-Network-Based C-Measure Map-Matching Algorithm for Car Navigation System," IEEE, vol. 48, no. 2, 2002, pp. 432-441.

[9] J. Byeong-Chan and S. Kim, “Multipath interference cancellation technique for high precision tracking in GNSS receiver,” vol.93, no.7, 2010, pp.1961-1964.

[10] N. Kubo, S. Kondo, and A. Yasuda, “Evaluation of code multipath mitigation using a software GPS receiver,” vol.88, no.11, 2005, pp.4204-4211.

[11] S. Kim, J. Byeong-Chan, and S. Lee, “DoA estimation of line of sight signal in multipath channel for GNSS receiver,” vol.92, no.11, 2009, pp.3397-3400.

[12] S.J. Hwan, J. Heo, S. Yoon, and K.S. Young, “Interference cancellation and multipath mitigation algorithm for GPS using subspace projection algorithms,” vol.91, no.3, 2008, pp.905-908.

[13] G. M. Djuknic and R. E. Richton, “Geolocation and assisted GPS,” vol.34, no.2, 2001, pp.123-125.

[14] C. Brenner and B. Elias, "Extracting landmarks for car navigation systems using existing GIS databases and laser scanning," ISPRS Archivers, vol. XXXIV, part3/W8, 2003, pp. 131-136.

[15] M. S. Braasch, "GPS Receiver Architectures and Measurements," IEEE, vol. 87, no. 1, 1999, pp. 48-64.

[16] P. Misra, and P. Enge, "Global Positioning system: signals, measurements, and performance," Ganga-Jamuna Press, 2001.

[17] X. Meng, G. W. Roberts, A. H. Dodson, E. Cosser, J. Barnes, and C. Rizos, "Impact of GPS satellite and pseudolite geometry on structural deformation monitoring: analytical and empirical studies," Journal of Geodesy, vol. 77, no. 12, 2004, pp. 809-822, doi:10.1007/s00190-003-0357-y.

[18] T. Sakai, "GPS notameno Zitsuyo Programing (Practical Programing for GPS)," Tokyo Denki University Press, 2007.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
issn: 1942-2679

International Journal On Advances in Internet Technology
issn: 1942-2652

International Journal On Advances in Life Sciences
issn: 1942-2660

International Journal On Advances in Networks and Services
issn: 1942-2644

International Journal On Advances in Security
issn: 1942-2636

International Journal On Advances in Software
issn: 1942-2628

International Journal On Advances in Systems and Measurements
issn: 1942-261x

International Journal On Advances in Telecommunications
issn: 1942-2601