International Journal on Advances in Networks and Services

2019 vol. 12 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 12, Number 3 & 4, 2019

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, Deloitte Consulting 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
Orhan Dagdeviren, International Computer Institute/Ege University, Turkey
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 / University of Plovdiv “Paisii Hilendarski”, Bulgaria
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
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
Pascal Lorenz, University of Haute Alsace, France
Albert A. Lysko, Council for Scientific and Industrial Research (CSIR), South Africa
Pavel Mach, Czech Technical University in Prague, Czech Republic
Elsa María Macías López, University of Las Palmas de Gran Canaria, Spain
Damien Magoni, University of Bordeaux, France
Ahmed Mahdy, Texas A&M University-Corpus Christi, USA
Zoubir Mammeri, IRIT - Paul Sabatier University - Toulouse, France
Gianfranco Manes, University of Florence, Italy
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Moshe Timothy Masonta, Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa
Hamid Menouar, QU Wireless Innovations Center - Doha, Qatar
Guowang Miao, KTH, The Royal Institute of Technology, Sweden
Mohssen Mohammed, University of Cape Town, South Africa
Miklos Molnar, University Montpellier 2, France
Lorenzo Mossucca, Istituto Superiore Mario Boella, Italy
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Katsuhiro Naito, Mie University, Japan
Deok Hee Nam, Wilberforce University, USA

Sarmistha Neogy, Jadavpur University- Kolkata, India
Rui Neto Marinheiro, Instituto Universitário de Lisboa (ISCTE-IUL), Instituto de Telecomunicações, Portugal
David Newell, Bournemouth University - Bournemouth, UK
Ngoc Tu Nguyen, Missouri University of Science and Technology - Rolla, USA
Armando Nolasco Pinto, Universidade de Aveiro / Instituto de Telecomunicações, Portugal
Jason R.C. Nurse, University of Oxford, UK
Kazuya Odagiri, Yamaguchi University, Japan
Máirtín O'Droma, University of Limerick, Ireland
Jose Oscar Fajardo, University of the Basque Country, Spain
Constantin Paleologu, University Politehnica of Bucharest, Romania
Eleni Patouni, National & Kapodistrian University of Athens, Greece
Harry Perros, NC State University, USA
Miodrag Potkonjak, University of California - Los Angeles, USA
Yusnita Rahayu, Universiti Malaysia Pahang (UMP), Malaysia
Yenumula B. Reddy, Grambling State University, USA
Oliviero Riganelli, University of Milano Bicocca, Italy
Antonio Ruiz Martinez, University of Murcia, Spain
George S. Oreku, TIRDO / North West University, Tanzania/ South Africa
Sattar B. Sadkhan, Chairman of IEEE IRAQ Section, Iraq
Husnain Saeed, National University of Sciences & Technology (NUST), Pakistan
Addisson Salazar, Universidad Politecnica de Valencia, Spain
Sébastien Salva, University of Auvergne, France
Ioakeim Samaras, Aristotle University of Thessaloniki, Greece
Luz A. Sánchez-Gálvez, Benemérita Universidad Autónoma de Puebla, México
Teerapat Sanguankotchakorn, Asian Institute of Technology, Thailand
José Santa, University Centre of Defence at the Spanish Air Force Academy, Spain
Rajarshi Sanyal, Belgacom International Carrier Services, Belgium
Mohamad Sayed Hassan, Orange Labs, France
Thomas C. Schmidt, HAW Hamburg, Germany
Véronique Sebastien, University of Reunion Island, France
Jean-Pierre Seifert, Technische Universität Berlin & Telekom Innovation Laboratories, Germany
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
Cristian Stanciu, University Politehnica of Bucharest, Romania
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

Rui Teng, Advanced Telecommunications Research Institute International, Japan
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 12, Numbers 3 & 4, 2019

CONTENTS

Cooperative V2X for Cluster-Based Vehicular Networks

Yongyue Shi, Xiao-Hong Peng
School of Engineering & Applied Science
Aston University
Birmingham, UK
Email: {shiy9, x-h.peng}@aston.ac.uk

Guangwei Bai
Department of Computer Science & Technology
Nanjing Tech University
Nanjing, China
Email: bai@njtech.edu.cn

***Abstract*— Timely data services supported by efficient vehicular communications are essential for connected and autonomous vehicles and future transportation systems. In this paper, a cluster-based two-way data service model is introduced to promote efficient cooperation between Vehicle-to-Vehicle and Vehicle-to-Infrastructure communications, or namely V2X, so that the service delivery performance across the vehicular network can be improved. Our results show that the cluster-based model can significantly outperform the conventional non-cluster schemes, in terms of service ratio, network throughput and energy cost.**

Keywords*— *V2X communications; clustering; service delivery model; energy cost.

I. Introduction

The main challenge in the development of smart mobility and intelligent transport systems is how to effectively manage traffic congestion, reduce car accidents and energy consumption with the rapidly increasing number of vehicles and complex road networks. It is vital to make traffic information (e.g., speed and vehicle density) and environmental information (e.g., weather and road condition) timely available for road users and network operators, to ensure road safety and traffic efficiency [1].

The Vehicular Ad-hoc Network (VANET) is an extended version of the Mobile Ad-hoc Network (MANET) and intended for improving driving safety and efficiency through both vehicle-to-vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. V2V and V2I can be operated cooperatively as V2X, making the VANET play a better role in modern transportation systemes under a complex traffic environment.

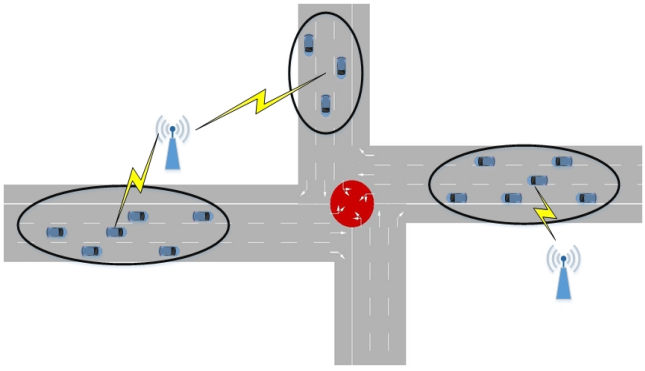


Figure 1. A VANET model with clusters

This paper proposes a V2X-based service system where the clustering technique is applied to improve transmission and energy efficiencies by significantly reducing the number of V2I connections. A cluster is a group of vehicles within the transmission range of each other, as shown in Fig. 1 where cluster heads exchange data with RSU via V2I while the other cluster members communicate with cluster heads via V2V. A data service model with cooperative V2X transmission via clustering is also introduced in this work, for effectively uploading the local information to the database and downloading the required service data from RSUs.

The remaining of the paper is organized as follows. Related work is discussed in Section II. Following the clustering algorithm presented in Section III, the proposed data service model and an energy model for performance analysis are presented in Section IV. Section V explains the simulation results produced by OMNET++, SUMO and MATLAB software tools. Finally, Section V concludes the paper.

II. Related Work

The idea of combining V2I and V2V has been applied in many works on VANET. In [2], Noori et al. explore the combination of various forms of communication techniques, e.g., cellular network, Wi-Fi and ZigBee for VANETs. In [3], a roadside unit (RSU) plays a vital part to provide services and make scheduling arrangements using a simple network coding in a V2X approach. This approach may cost more energy to complete the service and does not consider the packet loss and associated latency caused by the failed services. In [4], multiple RSUs are involved in broadcasting data periodically to vehicles via V2I and forwarded to vehicles via V2V if they are outside the transmission range. This model requires efficient handover mechanisms to ensure stable and in-time data services between the vehicles concerned.

The Dedicated Short-Range Communications (DSRC) technology refers to a suite of standards of Wireless Access in Vehicular Environments (WAVE) [5] and supports both V2V and V2I communications. Vehicles equipped with sensors can collect local traffic and environment information and exchange it for the similar information of other regions (place of interest) with RSUs. A RSU acts as an interface between vehicles and the vehicular network to provide vehicles the service information requested and pass on the collected information to other parts of the network. The high mobility

and density of vehicles presents a big challenge in V2X communications, which causes congestion in service delivery in this environment. In addition, moving vehicles will keep exchanging information between them and this will cost a significant amount of energy for continuous data sensing, transmission and processing, especially for V2I as it needs to cover longer distances than V2V.

The Lowest-ID clustering algorithm is a basic method to select cluster head, which uses unique vehicle ID numbers as the selection standard [6]. This algorithm works stably in most MANETs but may not always be suitable for VANET due to higher velocity and more restricted routes for vehicles. The AMAC (Adaptable Mobility-Aware Clustering) algorithm [7] considers mainly the destination as the key factor in forming clusters to improve the stability of clusters and extend the cluster's lifetime. However, the destination may not always be collected from navigation systems as drivers do not always use them for the known routes. A three-layer cluster head selection algorithm based on the interest preferences of vehicle passengers is proposed for multimedia services in a VANET [8]. This scheme is inefficient when the requirements in operations differ too much.

Based on the discussion of V2X related work, a more efficient service delivery method is introduced in this paper by utilizing clusters and minimizing channel congestion caused by excessive V2I transmission in conventional service models. We will show that the cluster model outperforms the non-cluster model at both service and energy performance levels.

III. Clustering Algorithm

In MANETs, moving vehicles can be divided into different sizes of clusters, such as using the "combined weight" algorithm to select cluster heads [9]. The selection takes the current position, number of neighbours, mobility, and battery power of vehicles into consideration. In VANETs, vehicles' mobility is more limited by the road type, traffic signs and other traffic factors. Therefore, the elements involved in forming clusters in a VANET need to be adjusted accordingly.

Clustering in VANET needs to be adjusted in accordance with traffic features such as high mobility of vehicles, regular moving tracks and directions. The clustering algorithm for forming and maintaining clusters should ideally be stable and adaptive to vehicle mobility or sudden changes in network topology, and provide reliable end-to-end communications across the network. The algorithm presented here is focused at the methods for forming a cluster including cluster head selection and cluster operation, which are discussed below.

A. Cluster Head Selection

Clustering in VANET is an extension of clustering in MANET, where the mobility and channels show different features. On the road networks, vehicles equipped with communication devices group themselves into clusters according to certain rules applicable to the road environment and traffic characteristics. Cluster size is not fixed and usually depends on the communication range of vehicles and the traffic environment.

There are three types of nodes (vehicles) in a VANET: Free Node (FN), Cluster Head (CH), Cluster Member (CM). The clustering algorithm considers the one-hop neighbours of each node and the cluster size is decided by cluster head's communication range. CH is responsible for collecting data and service requests from CMs, uploading current driving information (e.g., traffic is normal or congested), and requesting services from the RSUs. This paper defines a new weighting metric for selecting the CH, considering the factors, such as position, velocity, connectivity and driving behavior of the vehicles involved.

The position of each node is obtained from GPS (Global Positioning System) data. The average distance, P_i, between CH and CM should be as short as possible, which is given by

$$P_i = \frac{1}{n}\sum_{j=1}^{n}\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (1)$$

where n is the number of neighbors of node n_i, x and y are coordinate values of two nodes involved.

The velocity of CH, V_i, is defined to be the difference between the velocity of a candidate node v_i and the average velocity for the current traffic flow, and given by:

$$V_i = \left| v_i - \frac{1}{n}\sum_{j=1}^{n} v_j \right| \quad (2)$$

where v_j is the velocity of the j-th neighbour of the candidate node.

The connectivity of the candidate node is reflected by the number of its neighbors, N_i. The ideal connectivity is denoted as σ, which represents the maximum number of neighboring nodes within one hop without causing traffic congestion, and is given as:

$$\sigma = 2R_t \times 133 \times n_l / 1000 \quad (3)$$

where R_t is the transmission range, n_l is the number of lanes. The constant value 133 represents the highest possible density (vehicles/(lane·km) [10]. The actual connectivity, C_i, is used to measure how close the N_i is to the ideal value σ, i.e.:

$$C_i = \left| N_i - \sigma \right| \quad (4)$$

The last factor is the acceleration of the vehicle, a_i, to reflect the driving behaviour D_i by showing how stable a vehicle is when running along the road, i.e.:

$$D_i = \left| a_i \right| \quad (5)$$

The weighting matrix is formed by combining the four factors, discussed above, which are considered equally important. After the normalization of the four measurements, as shown below,

$$\text{a) } P_i' = \frac{P_i}{P_{\max}}, \text{ b) } V_i' = \frac{V_i}{V_{\max}}, \text{ c) } C_i' = \frac{C_i}{\sigma}, \text{ d) } D_i' = \frac{D_i}{D_{\max}} \quad (6)$$

the weighting matrix, W_i, is defined as

$$W_i = P_i' + V_i' + C_i' + D_i' \quad (7)$$

where P_{max} is the distance between the i-th vehicle and the farthest vehicle from it, V_{max} is the speed limitation by traffic rules that a vehicle can reach in the flow, D_{max} is the maximum absolute value of acceleration the vehicle can reach when it is running. A smaller W_i indicates the higher suitability of the candidate for the CH.

B. Cluster Operation

In the environment where vehicles have high mobility, cluster stability is a key factor to consider. In some traffic scenarios, in order to improve the transmission efficiency some vehicles are prevented from joining in a cluster. On a two-way straight road, for example, a vehicle is not allowed to join in the cluster that is running in the opposite direction. To maintain an established cluster in dynamic traffic and environmental conditions, different types of information packets are used for coordinating the operation with the cluster and delivering services, such as:

1) Vehicle Information Packet (VIP): It carries the basic vehicle information, including vehicle ID, velocity and position. VIP is used for starting the cluster forming process. When a vehicle detects itself as a free node (FN), it sends its VIP to its neighbours and enables them to calculate its weight value *Wi* based on (7), which is the basis of CH selection: the vehicle with the smallest *Wi* value becomes the CH.

2) Cluster Head Announcement (CHA): When a vehicle considers its weight low enough to be a CH, it will broadcast a CHA together with its weight value *Wi*. Other vehicles will compare the received *Wj* with their own weight and send their CHA to argue if theirs have a smaller weight than *Wi*.

3) Cluster Head Maintain (CHM): A node with the smallest *Wi* is elected as CH, and it then sends CHM to all its neighbours to declare its identity (CH ID). This packet is broadcast periodically if CH considers its status still suitable to be a cluster head.

4) Service Data Packet (SDP): SDP consists of two parts: head and context. The head includes the packet ID, sender ID and time stamp. The context part carries the actual communication message such as service requests and collected information.

To reduce the transmission cost, CH does not keep the list of its members, every CM stores the CH's ID to identify its cluster. When CH broadcasts the service packets, CMs who have the same CH ID and the targeted service ID will receive the service packets.

IV. Cooperative Service Model

A. Service Delivery

In the pure V2I service model, all vehicles are directly connected to RSUs for service delivery, resulting in transmission congestion due to a limited number of frequency channels and higher transmitting power to cover distance between vehicles and RSUs.

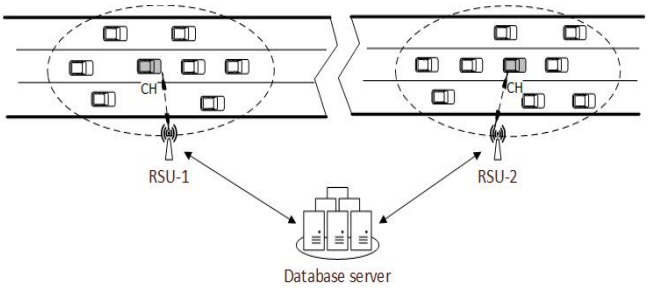


Figure 2. Cluster-based service model

The service model that we have developed utilizes cluster-based V2X communications. In this model, vehicles are grouped into clusters for information exchange between vehicles and RSUs. CHs are selected to gather and aggregate information collected by CMs and disseminate service packets to CMs via V2V. V2I transmissions take place only between CHs and RSUs via V2I directly, including uploading information to the server via RSU and downloading service data from RSU by CHs, as shown in Fig. 2.

This system model enables real-time information sharing. In addition, the cluster-based service model has transferred most of the data delivery from long-range V2I to short-range V2V. In this way, both transmission collision in the vehicle-RSU links and energy consumption can be reduced as RSUs only need to communicate with CHs. The database server shown in Fig. 2 stores service information including the traffic and environmental information such as the velocity of current traffic flow, real-time density of vehicles, weather conditions and road status, which is updated periodically.

This service system follows the standards of IEEE 802.11p and IEEE 1609 family [5], which specifies 7 channels of 10 MHz with each including one control channel and 6 service channels. The transmitting power levels are up to 44.8dBm for RSUs and 33dBm for vehicles. The control channel is used for exchanging control messages and safety information, while service channels are used for delivering service information packets.

Both uplink and downlink transmissions of the proposed service model are described as below.

B. Uplink Transmission

The vehicles on roads have different regions of interest and tend to learn the environmental and traffic conditions in those regions in advance. They also collect current traffic information from their on-board sensors and upload it to the road network for traffic control. Upon generating a message with a high priority (e.g., an accident), it is their responsibility to report it to CH. Each vehicle generates packets including vehicle ID, request ID and CH ID. Every CM submits the requests with the collected information to CH and then set the timer to wait for services. On receiving the packets, CH aggregates the collected information and forward to RSU in uplink transmission.

The traffic/environmental information includes the average speed of current flows, position, weather (rain, fog, lights etc.) and traffic conditions (smooth or congested etc.).

Vehicles within the same cluster may gather similar information, especially the weather and road conditions. In addition, different vehicles may request information for the same regions. Therefore, CH integrates the collected information before forwarding it to RSU. The aggregated data at CH will be less than what it has been collected, so the transmission efficiency of uplink transmission can be improved. Emergent messages (e.g., accidents alerts) will be marked with a higher priority during data aggregation.

Each RSU maintains its own database to store the recent service information collected from different CHs within its coverage. RSUs in different areas will periodically exchange and update information between them. In this case, vehicles in one area can learn the information about a larger range of areas ahead. The information service helps drivers to choose the best routes to reach their destinations and avoid congestion and accidents. They can also be aware of the travelling time they will spend.

C. Downlink Transmission

Upon receiving the packets from CH, RSU updates the database with collected information and generates the service packets requested by vehicles. These packets are sent to CH in downlink transmission via V2I and CH will redistribute them to CMs via V2V. Once overhearing the corresponding service ID and CH ID, CM will store the packets and mark the received request as satisfied. If CM is not satisfied (i.e., CM did not receive the requested service data) during a waiting time threshold, this request is considered as failed. In this case, a new request will be generated and sent to its CH again.

It is likely that a service is requested by multiple vehicles (e.g., three cars are interested in the traffic information of the same area), so CH disseminates data to CMs by broadcasting to help reduce the transmission cost. When CH broadcasts service packets, the CMs that are marked with the same CH ID and service ID will receive and save the service packets sent by Ch. Our model can provide real-time services for vehicles and allow drivers to manage their routes and time efficiently.

In addition, through clustering most of the data delivery between RSU and vehicles (i.e., V2I) is now transformed to data exchange between cluster members and cluster head (i.e., V2V). In this way, both transmission collision and energy consumption at the RSUs level can be reduced.

This service model follows the standards of IEEE 802.11p and IEEE 1609 family, which specifies 7 channels including one control channel and 6 service channels. Each channel has 10 MHz. The control channel is used for exchanging control messages and safety information, while service channels are used for delivering service information packets.

D. Energy Model

The proposed service model groups vehicles into clusters to improve the service efficiency and reduce energy consumption. Within a cluster we only consider one-hop transmission between neighboring vehicles to ensure the stability of connections between them. The energy model covers transmission in both directions, i.e., uplink and downlink shown in Fig. 3.

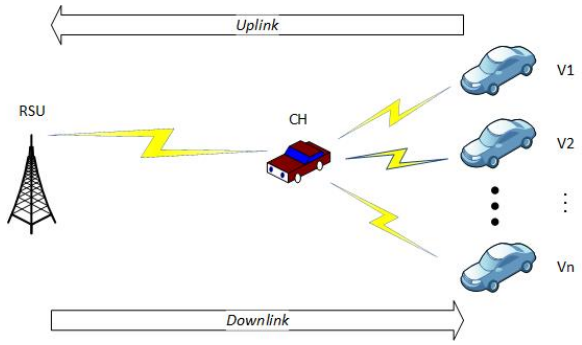


Figure 3. Uplink and downlink scenarios.

For comparison purposes, we consider two different service models: one is to provide services based on clusters (V2X) and the other is to provide services without clusters (pure V2I). In our service model using cluster-based V2X, both V2V and V2I are used for data uploading in the uplink and providing services in the downlink. However, higher transmit power is required in the V2I mode than in the V2V mode as the distances between RSU and vehicles are generally much longer than the distances between vehicles themselves within a cluster. In this subsection, we will discuss the energy performances of the cluster-based service model in comparison with the non-cluster service model where services are delivered through V2I only.

The RSU is up to 8-15 meters high [11] and the distance between a RSU and vehicles is much farther than the distance between vehicles themselves, thus V2I requires higher transmitting power than V2V to deliver data. The transmitting power for V2V mainly depends on the distance between CH and the farthest CM from CH and the maximum transmission distance (d^*) in this case is mainly based on the number of vehicles in a cluster. Denoting the distance between two vehicles as $d_{i,j}$, then:

$$d^* = \max_{i,j \in n} \{d_{i,j}\} \tag{8}$$

With the following definitions: P_{tvi}, - the transmission power of the i-th transmission by a vehicle to another vehicle; P_{tri} - the transmission power of the i-th transmission by a vehicle to a RSU; P_{rv} - the lowest receive power at another vehicle; P_{rr} - the lowest receive power at RSU; L_{pvi} - the path loss of a transmission link between two vehicles; and L_{pr} - the path loss of a transmission link between the vehicle and the RSU, the minimal required transmitting power for this vehicle is:

$$P_{tvi} = P_{rv} \cdot L_{pvi} \tag{9}$$

$$P_{tri} = P_{rr} \cdot L_{pri} \tag{10}$$

where the two path losses in (9) and (10) are defined by (using the two-ray model [12]):

$$L_{pvi} = \frac{d_{vi}^{\ 4}}{h_{vt}^{\ 2} h_{vr}^{\ 2} G_{vt} G_{vr}} \tag{11}$$

$$L_{pri} = \frac{d_{ri}^{\ 4}}{h_{vt}^{\ 2} h_{rr}^{\ 2} G_{vt} G_{rr}} \tag{12}$$

where G_{vt}, G_{vr} and G_{rr} are the antenna gains of the transmitting vehicle, the receiving vehicle and the RSU, respectively; h_{vt}, h_{vr} and h_{rr} are the antenna heights of the transmitting vehicle, the receiving vehicle and the RSU, respectively; d_{vi} and d_{ri} are the distance between vehicles and the distance between the vehicle and the RSU, respectively.

In the non-cluster service model, only V2I communications take place in both uplink and downlink. Therefore, the total transmitting power, P_{tn}, comprises of the power used for uplink transmission, P_{up}, and power for downlink transmission, P_{down}, i.e.:

$$\begin{aligned} P_{tn} &= P_{up} + P_{down} \\ &= N\sum_{i=1}^{N_{tv}} P_{tvi} + N\sum_{i=1}^{N_{tr}} P_{tri} \end{aligned} \tag{13}$$

where P_{tvi} is the uplink transmitting power of a vehicle for the *i-th* transmission, P_{tri} is the downlink transmitting power of a RSU for the *i-th* transmission, and N is the total number of vehicles that communicate with RSU via V2I, N_{tv} is the total number of uplink transmissions and N_{tr} is the total number of downlink transmissions.

We assume that the transmission time spent on each uplink transmission is unchanged, i.e., t_{tv} and P_{tvi} is constant for all uplink transmissions. Similarly, t_{tr} is defined as the transmission time for each downlink transmission and P_{tri} is constant for all downlink transmissions. Therefore, the total energy consumed in the non-cluster service model is given by:

$$E_n = P_{up} t_{tv} + P_{down} t_{tr} \tag{14}$$

In our proposed model, cluster head aggregates the collected data from each member and these data may be of high similarity when cluster members are in a similar environment but may also differ from each other because of the unique requirement from each vehicle. This leads to different transmission times spent in V2I communications, depending on the level of data aggregation carried out by the cluster head. The similarity level of the data from different cluster members will affect the data size for CH to transmit to RSU. In addition, only the cluster head involves V2I communications.

In the cluster-based model, the uplink transmitting power comprises of power for V2V (CMs to CH), P_{utv}, and power for V2I (CH to RSU), P_{utr}, while the downlink transmitting power is also divided into two parts, i.e., power for V2V (CH to CMs), P_{dtv}, and power for V2I (RSU to CH), P_{dtr}. The total transmitting power, P_{tc}, in the proposed service model can be calculated as:

$$\begin{aligned} P_{tc} &= P_{up} + P_{down} \\ &= [P_{utv} + P_{utr}] + [P_{dtv} + P_{dtr}] \\ &= [(N-1)\sum_{j=1}^{N_{utv}} P_{utvj} + \sum_{j=1}^{N_{utr}} P_{utrj}] \\ &+ [(N-1)\sum_{j=1}^{N_{dtv}} P_{dtvj} + \sum_{j=1}^{N_{dtr}} P_{dtrj}] \end{aligned} \tag{15}$$

where N is the number of vehicles in a cluster, P_{utvj} is the transmitting power of the *j-th* transmission of a CM to the CH (uplink V2V), P_{utrj} is the transmitting power of the *j*-th transmission of the CH to the RUS (uplink V2I), P_{dtvj} is the transmitting power of the *j-th* transmission of the CH to a CM (downlink V2V), P_{dtrj} is the transmitting power of *the j-th* transmission of the RUS to the CH (downlink V2I), N_{utv}, N_{utr}, N_{dtv}, and N_{dtr} are the total numbers of uplink V2V, uplink V2I, downlink V2V and downlink V2I transmissions, respectively.

We also assume that an equal transmission time is spent on each of N_{utv} transmissions and is represented by t_{utv}. This assumption applies also to N_{utr}, N_{dtv}, and N_{dtr}, and represented by t_{utr}, t_{dtv} and t_{dtv}, respectively. Therefore, the total energy consumed in the cluster-based service model is given by

$$E_c = P_{utv} t_{utv} + P_{utr} t_{utr} + P_{dtv} t_{dtv} + P_{dtr} t_{dtr} \tag{16}$$

In uplink V2I transmissions, the transmission time will be reduced as result of data aggregation at the CH. For this reason, t_{utr} is scaled down from the original time duration required for transmitting all the data collected by CH from CMs, which is t'_{utr}, by applying a scaling factor W_s, such that

$$t_{utr} = t'_{utr} W_s \qquad (0 \le W_s \le 1). \tag{17}$$

E. Performance Evaluation

In this paper, the following four metrics are applied to evaluate the performance of the proposed system.

- Service ratio (γ). It is the ratio of the number of successful delivered requests n_s to the total number of requested services n. This is a vital metric to evaluate the effectiveness of the V2X system. This performance metric is given by:

$$\gamma = \frac{n_s}{n} \tag{18}$$

- Average service delay (τ). It is defined as the average duration from a vehicle submitting a service request to it finally receiving the service packets, which is expressed by:

$$\tau = \frac{\sum_{i=1}^{n_s} t_{si} + n_{us} \cdot t_p}{n_s} \tag{19}$$

where t_{si} is the time duration of the i-th successful service transmission, n_{us} is the number of unsuccessful

service requests, and t_p is the waiting time a vehicle spends for the service that is not delivered.

- Throughput (η). It is a widely applied metric to evaluate the transmission efficiency of a system. It is defined as the average size of data successfully delivered over a time unit.

$$\eta = \frac{p_s}{T} \tag{20}$$

where p_s is the total size of delivered service packets, T is the total transmission time.

- Energy Cost (E_C). It is measured as an average amount of energy (Joule) consumed for transmitting one bit of data, or called energy per bit. Given transmitting power P_t and throughput η, the energy cost is given by

$$E_C = \frac{P_t}{\eta} \tag{21}$$

where p_s is the total size of delivered service packets, T is the total transmission time.

Energy cost can also be represented by a ratio of the energy consumption E to the amount of data B transmitted by consuming E Joules.

The energy cost in the uplink (E_{Ub}) is defined as:

$$E_{Ub} = \frac{E_U}{B_U} = \frac{E_{UV} + E_{UI}}{B_{UI}} \tag{22}$$

where B_{UI} is the size of data transmitted in the uplink. B_{UI} is determined by the data loss rate and the aggregation degree, which is shown as below:

$$B_{UI} = (1 - P_{UI}) \cdot B_A \tag{23}$$

$$B_A = (1 - P_{UV}) \cdot (A_I \cdot B_{UV}) \tag{24}$$

where P_{UV} and P_{UI} are the data loss rates in V2V and V2I transmissions in the uplink, respectively; B_A is the aggregated data size, B_{UV} is the size of data transmitted from the vehicles via V2V; and A_I is the aggregation degree, defined as:

$$A_I = \frac{j}{n'} \tag{25}$$

where, $j = 1, 2, \ldots n'$, n' is the number of CMs whose data are successfully received by CH.

The energy cost in the downlink (E_{Db}) has a similar expression to that for the uplink, except there is no need of aggregation as no duplicated data is sent out from RSU, which is represented by:

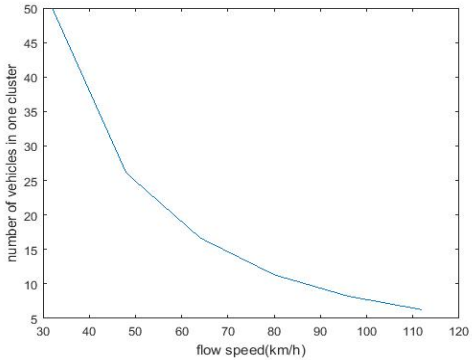


Figure 4. Service ratio under different flow speeds

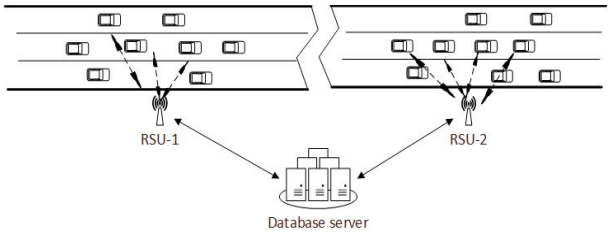


Figure 5. Non-cluster service model

$$E_{Db} = \frac{E_D}{B_D} = \frac{E_{DV} + E_{DI}}{B_{DV}} \tag{26}$$

where B_{DV} and B_{Di} are the sizes of data transmitted by V2V and V2I in the downlink, which are given by:

$$B_{\mathrm{DV}} = (1 - P_{DV}) \cdot B_{Di} \tag{27}$$

$$B_{\mathrm{Di}} = (1 - P_{Di}) \cdot B_D \tag{28}$$

where P_{DV} is the data loss rate during V2V in the downlink, P_{Di} is the data loss rate during V2I in the downlink.

Overall, the energy cost of the whole system is:

$$E_C = \sum_{i=1}^{n_U} E_{Ub} + \sum_{j=1}^{n_D} E_{Db} \tag{29}$$

where n_U is the total number of uplink transmissions and n_D is the total number of downlink transmissions.

V. SIMULATION AND RESULTS ANALYSIS

A. Simulation Setup

The traffic scenarios and communications models are simulated using SUMO [13] and OMNET++ [14]. SUMO is a powerful traffic simulator and supports multiple road topologies and vehicle attributes. It can cooperate with other network simulators via its Traffic Control Interface (TraCI) modules. OMNET++ is an extensible, modular, and

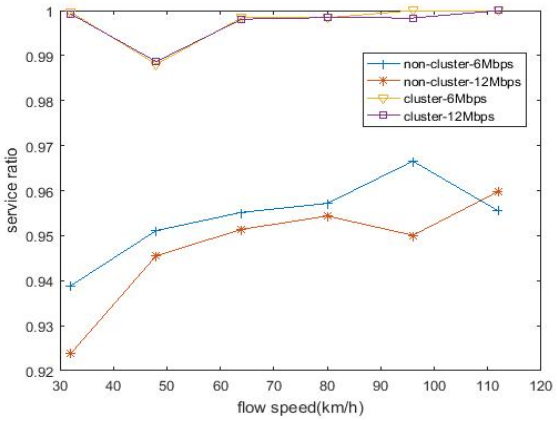


Figure 6. Service ratio under different flow speeds

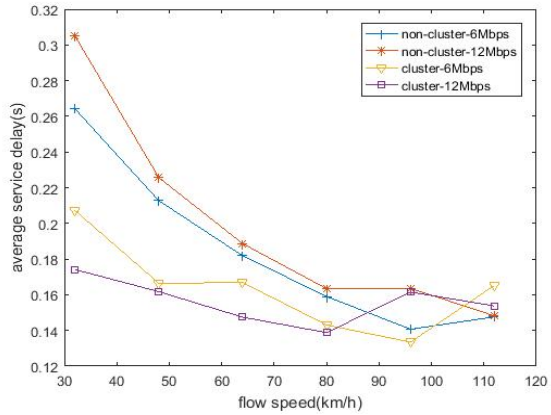


Figure 7. Average service delay under different flow speeds

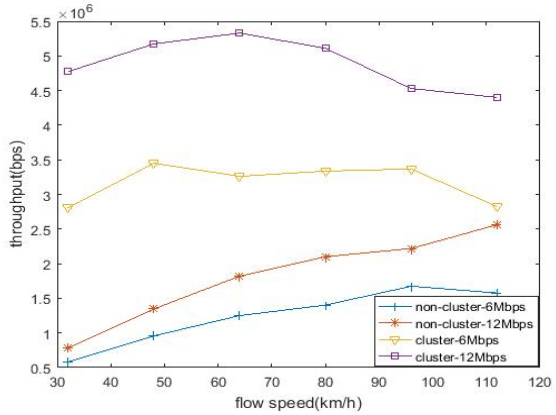


Figure 8. Throughput under different flow speeds

TABLE I. SIMULATION PARAMETERS

Parameters	Value
Frequency band	5.850-5.925 GHz
Channel bandwidth	10 MHz
Receive power sensitivity	-89dBm
Propagation model	Free space model
Data rate	6Mbps, 12Mbps
Number of requests	20-25
Data size	1000 bits
Number of lanes	3
Simulation time	300s

component-based C++ simulation framework, supporting various types of network simulation developments.

We built a one-way straight road with three lanes on SUMO, in which vehicles in each lane are running as a flow and the related service model is shown in Fig. 2. According to the Highway Code [15], the safe stopping distances are related to the driving speed. Considering the transmission range of V2V, which is usually 300 metres, the number of vehicles in a cluster on motorways is related to the flow speed as well.

Based on the safe stopping distance, we define six scenarios in simulation for the flow speed of 32, 48, 64, 80, 96, 112 km/h, respectively. The relationship between the vehicle number and flow speed is shown in Fig. 4.

The transmission model is configured based on the IEEE 802.11p and IEEE 1609 Family. Table I gives the parameters of the physical and MAC (Media Access Control) layers of the vehicular communication system and Table II specifies the transmission power in different modes (V2V and V2I), which are adopted in simulations.

For the purpose of performance comparison, we have also simulated the non-cluster model, as shown in Fig. 5 where the same number of vehicles and vehicle velocity are set in each scenario. Once the vehicles enter the transmission range of the RSU, they communicate with RSU directly via V2I. The two models are evaluated over the same set of performances, featuring the service ratio, average service delay, throughput and energy consumption.

TABLE II. TRANSMISSION POWER IN V2V AND V2I

Flow speed *(km/h)*	32	48	64	80	96	112
V2V *(mW)*	0.802	1.020	0.899	0.867	0.925	0.711
V2I *(mW)*	2.885	2.898	2.890	2.841	2.878	2.821

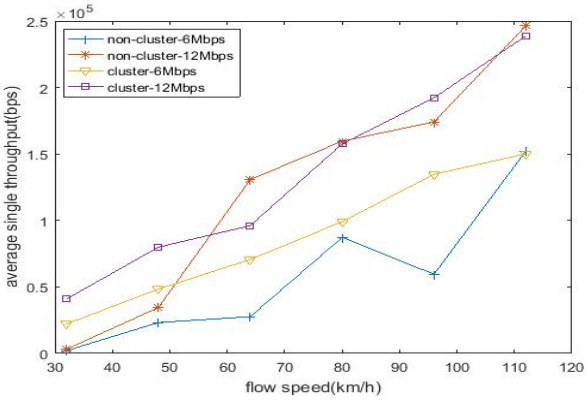


Figure 9. Average throughput of individual vehicles under different flow speeds

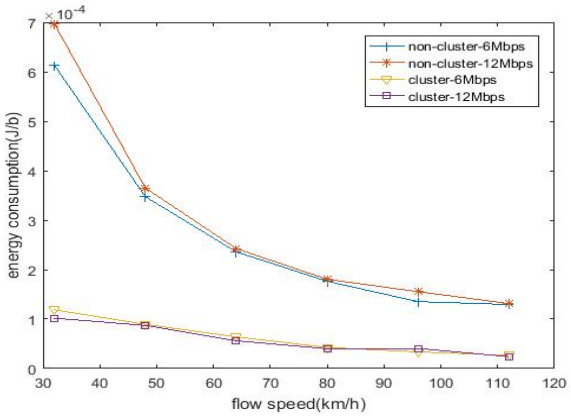


Figure 10. Energy cost under different flow speeds

B. Results analysis

Fig. 6 shows different service ratios (or successful rate of service delivery) of both Cluster-Based (CB) and Non-Cluster (NC) service models under 6 different scenarios and with different flow speeds and vehicle densities. CB achieves higher and more stable service ratios than NC under all scenarios and at both 6Mbps and 12Mbps data rates. The service ratio of NC also shows a raising trend with the increase of the flow speed. This is due to the lower vehicle density when the flow speed is higher, which reduces transmission collision and congestion.

When the flow speed is low, the distance between vehicles is relatively short and more vehicles are involved within the same transmission range, leading to more service requests and local data collected for transmission. In this scenario, by grouping vehicles into clusters, transmission loads between vehicles and RSUs are reduced, hence less collision events in the CB model than in the NC model.

When vehicles move out of the transmission range of RSU, those without support of clusters will not be able to receive service packets directly from RSU. But in the cluster-based model, CMs can still obtain services from the CH that has stored service data from RSU as long as they are in the transmission range with the CH via V2V.

The average service delay is shown in Fig. 7, which includes the time spent on transmitting service data and the waiting time for re-transmission when the previous service delivery is failed. In the NC model, each vehicle has to wait for downloading service data from the RSU in turn. This delay is reduced in the CB model since only CH is involved in V2I transmissions.

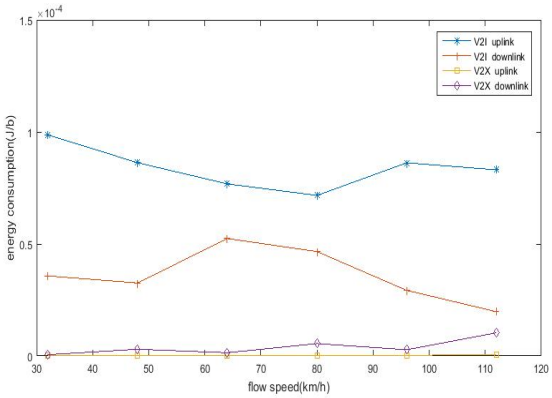


Figure 11. Energy cost in uplink and downlink.

In addition, more time can be saved by using a cluster where CH transmits aggregated sensing data collected from CMs and broadcasts service data from RSU to the CMs that request the same information. The delay profile presented in Fig. 7 is also correlated with the service ratio results shown in Fig. 6.

When the flow speed increases, there will be less collision or congestion cases as a fewer number of vehicles are involved in transmission, thus in this scenario the CB model does not show as much advantages as they have at low flow speeds.

In Fig. 8, it is shown that the CB model clearly outperforms the NC model in terms of the network throughput under all six different scenarios. Throughput in the CB model appears to be more sustainable than that in the NC model, and the gaps between them are data rate dependant. As we can see, the CB's throughput at 6 Mbps is up to 2.3 times higher than that of the NC model, while when at 12 Mbps the difference is increased to up to 5 times. However, the throughput of the NC model also increases with the flow rate as less service requests are generated at high flow speeds or low vehicle densities.

The average throughput of individual vehicles is shown in Fig. 9 versus the flow speed. Generally, the throughput of individual vehicles in all schemes increases with the flow speed. As higher flow speeds correspond to lower vehicle densities according to Fig. 4, lower congestion in data traffic and, as a result, higher throughput will be expected in this situation.

In addition, it also correlates proportionally with the data rate as well. At low flow speeds, the CB model has a clear throughput advantage over the NC model because clustering helps to improve transmission efficiency. But the NC model can achieve competitively high throughput when the flow speed increases and with a higher data rate.

The energy performance in terms of Joule per bit is demonstrated in Fig. 10 for the two service models. Vehicles in a cluster exchange data with a RSU via V2X, i.e., V2V between themselves and V2I between CH and RSU, while

when clusters are not used all transmissions rely on V2I. This will make a significant difference in energy consumption between the two service models, as shown in Fig. 10.

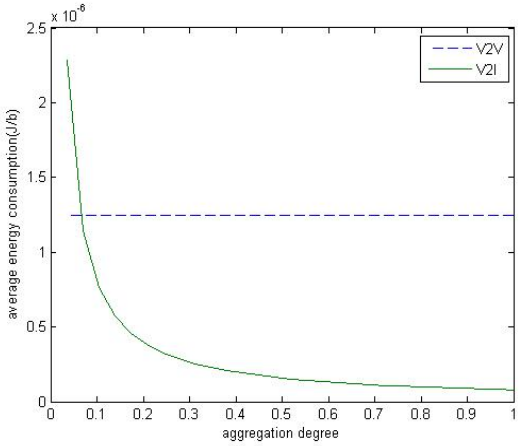
Like the results in other performance figures, the CB model is considerably more energy efficient than the NC model, and this advantage is particularly evident in the low flow-speed regions. The performance gap is closing down as the flow speed increases.

The energy performance can also be displayed separately in uplink as well as downlink, as shown in Fig. 11 where V2X refers to transmission mode in the clustered model while V2I represents transmission mode in the non-cluster model. Clearly, the clustered model is superior to the conventional non-cluster model as much lower energy cost is incurred in V2X than in V2I for both uplink and downlink transmissions.

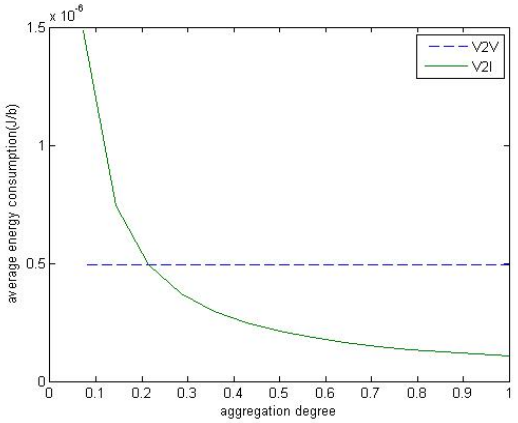
Fig. 12 shows the energy costs for both V2V and V2I (uplink) transmissions under varied aggregation degrees. The aggregation degree value '0' refers to no data aggregation at all as data from different vehicles are all different, and value '1' indicates that all the data from different vehicle are the same, so only one copy of data is needed for V2I transmission after aggregation. And for other values in between it means that data from different vehicles are similarity to some degree.

As shown in Table II, the transmitting power for V2I is much greater than that in V2V. But the energy cost in terms of energy per bit is affected by vehicle mobility (flow speed) and the similarity of data from different vehicles. Given the flow speed of 80 km/h in Fig. 12(a), the crossing point between V2V and V2I is when the aggregation degree A_I is fairly low (around 0.08). This means that V2I transmission can have a lower energy cost than V2V when A_I is greater than 0.08.

When the flow speed is increased to 112 km/h as shown in Fig. 12(b), the crossing point is moved to a higher value of 0.23. In this case, V2I could still be more energy efficient than V2V if more data can be removed through aggregation, i.e., having a higher aggregation degree.



(a) Flow speed 80km/h.



(b) Flow speed 112km/h.

Figure 12. Energy cost vs. aggregation degree at flow speed: (a) 80 km/h and (b) 112 km/h.

VI. CONCLUSION

In this paper, we propose a service delivery model via V2X in a vehicular network to improve the transmission efficiency and reduce energy costs. This model can effectively provide vehicles with real-time traffic and environmental information required for selecting the best routes to their destinations and avoiding traffic accidents or congestions.

A combined weighting metric is introduced in this paper and applied to the formation of clusters. The CH is selected based on the mobility and connectivity of vehicles to ensure the stability and efficiency of data exchange and service delivery. An energy model is also presented in this paper to provide an effective analytical tool for energy performance characterization comprising of both uplink and downlink transmissions.

As only CHs are responsible for direct communication with RSUs and dissemination of service data to other vehicles in the network, the cluster-based V2X approach presented in this work can significantly enhance service delivery efficiency and improve energy performance. This has been shown by simulation results, in terms of service ratio, average service delay, throughput and energy cost, in comparison with the performance of the V2I dominated non-cluster model.

Future work will consider more complicated scenarios in highway settings. The data aggregation method will be extended to develop specific data fusion and integration algorithms based on the information entropy theory. In addition, the two-way service model and associated energy analysis schemes will be established and investigated for developing a more realistic and optimized V2X service delivery platform.

REFERENCES

[1] Y.-Y. Shi, X-H. Peng, and G.-W. Bai, "Efficient V2X Data Dissemination in Cluster-Based Vehicular Networks," Proc. The 17th International Conference on Advances in Vehicular Systems, Technologies and Applications, June 2018.

[2] H. Noori and M. Valkama, "Impact of VANET-based V2X communication using IEEE 802.11 p on reducing vehicles traveling time

in realistic large scale urban area," Proc. International Conference on Connected Vehicles and Expo (ICCVE), 2013, pp. 654-661.

[3] K. Liu et al., "Network-coding-assisted data dissemination via cooperative vehicle-to-vehicle/-infrastructure communications," IEEE Trans. on Intelligent Transportation Systems, vol. 17, pp. 1509-1520, 2016.

[4] Q. Wang, P. Fan, and K. B. Letaief, "On the joint V2I and V2V scheduling for cooperative VANETs with network coding," IEEE Trans. on Vehicular Technology, vol. 61, pp. 62-73, 2012.

[5] Y. L. Morgan, "Notes on DSRC & WAVE standards suite: Its architecture, design, and characteristics," IEEE Communications Surveys & Tutorials, vol. 12, pp. 504-518, 2010.

[6] M. Jiang, J. Li, and Y. Tay, "Cluster Based Routing Protocol (CBRP). Draft-ietf-manet-cbrp-spec-01," txt, Internet Draft, IETF1999.

[7] M. M. C. Morales, C. S. Hong, and Y.-C. Bang, "An adaptable mobility-aware clustering algorithm in vehicular networks," Proc. Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific, 2011, pp. 1-6.

[8] I. Tal and G.-M. Muntean, "User-oriented cluster-based solution for multimedia content delivery over VANETs," Proc. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2012, pp. 1-5.

[9] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: A weighted clustering algorithm for mobile ad hoc networks," Cluster Computing, vol. 5, pp. 193-204, 2002.

[10] R. D. Kuhne, "Greenshields' legacy: Highway traffic," Transportation Research E-Circular, 2011.

[11] B. Association, Code of Federal Regulations, Title 47: Parts 80-End (Telecommunications): Federal Communications Commission Revised 10/05: Claitors Pub Div, 2006.

[12] A. Goldsmith, Wireless Communications, Cambridge University Press, pp.34-37, 2005.

[13] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-Simulation of Urban MObility," International Journal On Advances in Systems and Measurements, vol. 5, pp. 128-138, 2012.

[14] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," IEEE Trans. on Mobile Computing, vol. 10, pp. 3-15, 2011.

[15] The Highway Code, https://www.gov.uk/guidance/the-highway-code, 5th Dec. 2019.

Distributed Resilient Control Plane for Large Software Defined Networks

Eugen Borcoci, Stefan Ghita
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, stalghita@gmail.com

Abstract —**Large Software Defined Networks (SDN) solve the control scalability problem coming from the SDN control centralization principle, by defining and installing several regional controllers. Therefore, a controller placement problem (CPP) should be solved. During run-time, possible failures of links or node appear; then a forwarder node could try to select an available and reachable controller among those which are functional. This is called controller selection problem (CSP). Although many studies have been published, the above problems are still open research issues, given the various network contexts, providers' policies and possible multiple, different optimization criteria. Therefore, multi-criteria decision algorithms can provide valuable solutions. This paper is based on a previous work which has developed a simulation model for multi-controller SDN network, targeting optimization while including resilience aspects of the controller placement problem and controller selection problem. The current paper extends that work, by including a more in-depth analysis, giving relevant examples and introducing additional novel experiment results.**

Keywords — Software Defined Networking; Multi-criteria optimization; Controller placement; Controller selection; Forwarder nodes assignment; Reliability; Resilience

I. INTRODUCTION

This paper is an extended version of the work [1], which has been dedicated to study methods to optimize the *Software Defined Networking* (SDN) controller placement and selection in large area SDN networks. It is known that SDN has as basic principles the decoupling of the architectural *Control Plane* (CPl) w.r.t *Data Plane* (DPl) and also CPl centralization in SDN controllers. Therefore, in large network environments and considering the limited processing of controllers, scalability problems of the CPl appear [2]. The usual solution for this, adopted in many studies, is a distributed multi-controller implementation of the SDN control plane. Different flat or hierarchical organizations for a multi-controller SDN control plane have been developed, e.g., in [3][4].

In a basic approach, the *SDN controller* (SDN-C) is understood as a software control entity installed/placed in a geographically distinct location, i.e., a particular physical network node. The control plane is defined as an overlay network on top of the physical one. The links between controllers can be physical or virtual.

Recently, the *Network Function Virtualization* technologies [5] allow that several logical SDN-Cs could be realized as virtual entities running on top of virtual machines (notation for such controllers could be vSDN-C), i.e., several logical controllers can be collocated in the same physical node.

In this work we suppose the basic approach of the SDN controllers' implementation. However, the models developed in here can be as well applied to a virtualized environment. In such a case, the essential modification of the model (considering the optimization objective of this study) is that the logical controllers will be virtually linked through a control plane graph.

The *controller placement problem* (CPP) is a complex one, given the variety of factors involved. Some examples are: how the network topology is specified - flat or hierarchical/clustered; what criteria are considered to solve the CPP; number of controllers - predefined or not; failure-free or failure-aware metrics (e.g., considering backup controllers and node/link failures); how the DPl forwarders nodes are assigned/mapped to controllers (in static or dynamic way, i.e., depending on actual network conditions and network provider policies), and others. The evaluation of the degree of optimality of different approach can be studied on some simplified topologies – in order to compare the efficiency of approaches or, on real specific network topologies. Many studies, e.g., of Heller [6] et al. - as early study - and then others [7-9][11-19] considered various aspects and solutions of the CPP.

In a real network environment, it has been apparent that there is no unique best and universal placement rule for any SDN-controlled network. Dynamic nodes addition and deletion can happen and, in such cases, a forwarder could dynamically select an appropriate controller, if it has enough pertinent and updated information. The same situation appears when the traffic in the data plane is varying and re-assignment of forwarder to controllers is required, to avoid controller overload. This is *called controller selection problem* (CSP) and can be considered as an extension of the CPP [7].

The CPP was recognized as a non-polynomial (NP) - hard problem, mentioned in the early work of Heller et al. [6]. For such problems different approximation algorithms [8] and more pragmatic solutions have been proposed, adapted in different contexts. In particular, in SDN-controlled networks case, many optimization criteria have specific, target performance, both in the data plane and control plane, in failure-free or failure-aware approaches.

Examples of specific, individual criteria could be: to maximize the controller-forwarder or inter-controller communication throughput; reduce the latency of the path connecting them; limit the controller load imbalance; find an optimum controllers' placement and forwarder-to-controller allocation, offering a fast recovery after failures (controllers, links, nodes). Also, other specific optimization goals could be added to the above list, depending on specific context (wire-line, wireless/cellular, cloud computing and data center networks) and on some specific business targets of the Service Provider.

However, a major issue is that different optimization criteria could lead to significant different placement solutions; so, a multi-criteria global optimization could be a better trade-off approach.

The paper [9] provided a contribution on multi-criteria optimization algorithms for the CPP, not by developing specific single-criterion algorithms (many other studies already did that) but to achieve an overall optimization by applying *multi-criteria decision algorithms* (MCDA) [10]. The input of MCDA is the set of candidates (an instance of controller placement is called a candidate solution). Examples have been analyzed, on some real network topologies, proving the usefulness of the approach.

The more recent paper [1] extended the model of [9]; several reliability aware criteria have been added to the CPP solution. Also the novel CSP extension is introduced, being appropriate for a dynamic network context. It has been shown that the same basic MCDA can be applied in both static and dynamic context, but with different sets of criteria. Simulation experiments and novel results have been presented.

Note some limitations: neither work [9], nor [1] touch the problem of control plane overhead and signaling issues between the controllers when a re-configuration of the SDN network is performed. This could be the topic for additional studies. Also note that in this paper, by "large Software Defined Networks", it is actually understood networks having several SDN controllers.

The structure of this paper (extension of [1]) is described here. Section II is a short overview of related work. Section III revisits several metrics and optimization algorithms and presents some of their limitations. Section IV revisits the framework for MCDA-RL (the variant which is called "*reference level*") as a simple but powerful tool applicable to solve the CPP and CSP problems. Section V presents the implementation performed to validate the MCDA proposed model in a resiliency-oriented optimization approach, and outlines the simulation experiments performed. Section VI offers few examples of simulation results to illustrate the validity of the approach. Section VII presents conclusions and future work.

II. RELATED WORK

This short section is included mainly for guiding the reader to references. More comprehensive overviews on published work on CPP in SDN-controlled WANs are given in [11-14]. The main goal is to find those controller placements that provide high performance (e.g., low delay for controller-forwarder communications) and also create robustness to controllers and/or network failures.

An early work of Heller et al. [6] has shown that it is possible to find optimal controller placement solutions for realistic network instances, in failure-free scenarios, by analyzing the entire solution space, with off-line computations (the metric is latency). The studies [15-21] have been more focused, i.e., additionally considered the resilience as being important with respect to events like: *controller* failures, *network links/paths/nodes* failures, *controller overload* (load imbalance). The *Inter-Controller Latency* is also important and, generally, it cannot be minimized while simultaneously minimizing controller-forwarders latency; a tradeoff solution could be the answer.

The works [15][17] developed several algorithms for real topologies, aiming to find solutions for reliable. SDN control, but still keep acceptable latencies. The controller instances are chosen as to minimize connectivity losses; connections are defined according to the shortest path between controllers and forwarding devices. Muller et al. [18] eliminate some restrictions of previous studies, like: single paths, processing (in controllers) of the forwarders requests only *on-demand* and some constraints imposed on failover mechanisms. Hock et al. [16] adopted a multi-criteria approach for some combinations of the metrics (e.g., max. latency and controller load imbalance for failure-free and respectively failure use cases).

In a recent work [7], K. Sood and Y. Siang propose to extend the CPP problem into CSP, i.e., to consider the dynamics of the network and make controller selection. They explore the relationship between traffic intensity, resources requirement, and QoS requirements. It is claimed that to optimize the control layer performance, the solutions must be topology-independent and adaptive to the needs of the underlying network behaviour. They propose a topology independent framework to optimize the control layer, aiming to calculate the optimal number of controllers to reduce the workload, and investigate the placement/location of the controllers. However, their first declared objective has been not to determine the optimal placement of controllers in the network, but to motivate the CSP.

In recent papers [20][21] Y. Xu, M. Cello et al., developed dynamic forwarder/switch migration scenarios and algorithms, starting from a given switch-controller assignment and partition (based on some criteria) of the network in domains, each one controlled by a single controller. Also, a realistic assumption is considered, i.e., limited processing capacity of the controllers. During run time, if some controllers are overloaded (such events are dynamically observed by a monitoring system), then a heuristic algorithm is applied, to optimally move (re-assign) a part of the switches coordinated by that controller to other controller less loaded. In order to reduce the signaling (related to migration) between controllers, the migration is cluster-based, i.e., not a single switch is migrated but a cluster of switches are moved from an overloaded

controller, e.g., CT_i, to another less loaded controller CT_j. Thus, the algorithm realizes a controller load balancing (the name *BalCon* is coined for the algorithm [20]).

Many of the mentioned studies considered a single criterion in the optimization algorithms. In [9][1] a multi-criteria algorithm is used (applicable for an arbitrary number of decision criteria) to solve the CPP; validation of results have been presented for some real network topologies [22][23].

A recent work [24] studies the load balancing via switch migration in a network having several SDN controllers. The goal is to migrate switches from overloaded to under-loaded controllers, depending on the traffic variation. The work presents a heuristic approach to solve the switch migration problem. The advantage of the proposed solution versus other approaches is that the algorithm does not halt the search whenever a switch migration is not possible. Instead, it searches for more complex moves like swapping two switches to further improve the results.

The work in this paper is an extension of [1], with focus on optimal initial placement of the SDN controllers, considering among multi-criteria some reliability – related ones.

III. EXAMPLES OF CONTROLLER PLACEMENT METRICS AND ASSOCIATED ALGORITHMS

This section is a short presentation of a few typical metrics and optimization algorithms for CPP and CSP. A more detailed presentation of them can be found in [13]. Considering a particular metric (criterion) an optimization algorithm can be run for a given metric, as in [6][15-18].

As already stated, this paper goal is not to develop a new particular algorithm based on a given single metric, but to search for a global optimization. The individual metrics presented in this section can be embedded in a multi-criteria optimization algorithm.

The SDN-controlled network is abstracted by an undirected graph G(V, E), with V - set of nodes, E – set of edges and n=|V| the total number of nodes. The edges weights represent an additive metric (e.g., *propagation latency* [6]).

A basic metric is *d(v, c): shortest path* distance from a forwarder node $v \in V$ to a controller $c \in V$. We denote by C_i a particular placement of controllers; $C_i \subseteq V$ and $|C_i| < |V|$. 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 $C = \{C_1, C_2 \ldots\}$. Some metrics are basic, i.e., failure-free; others take into account failure events of links or nodes.

An important metric for SDN control is the *latency* between nodes. Note that, while it has a dynamic nature, in some simplified assumptions it is estimated as a static value.

A. Failure-free scenarios

- *Forwarder-to-controller latency*

In Heller's work [6], two (failure-free) metrics are defined for a given placement C_i: *Worst_case_latency* and *Average_latency* between a forwarder and a controller. In [5], the above two kinds of latencies are defined, for a particular placement C_i of controllers, where $C_i \subseteq V$ and $|C_i| \leq |V|$. The number of controllers is k for any particular placement C_i. The set of all possible placements is $C = \{C_1, C_2, \ldots.\}$. 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 minimum.*

The work [8] proposes an algorithm 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, but with an upper latency bound of each forwarder latency to its controller.

- *Inter-controller latency*

The SDN controllers should inter-communicate and therefore, the inter-controller latency is important. For a given placement C_i, one can minimize the *maximum latency between two controllers*. Note that this can increase the forwarder-controller distance (latency). Therefore, a trade-off is necessary, thus justifying the necessity to apply some multi-criteria optimization algorithms, e.g., like Pareto frontier - based ones [16].

B. Failure-aware scenarios

In such scenarios controller and/or network failures events are considered. The optimization process aims now to find trade-offs to preserve a convenient behavior of the overall system in failure cases (controllers, or nodes, or links).

- *Multiple-path connectivity metrics*

If multiple paths are available between a forwarder node and a controller [9], this can exploited in order to reduce the occurrence of controller-less events, in cases of failures of nodes/links. The goal in this case is to maximize connectivity between forwarding nodes and controller instances. A special metric can be defined as:

$$M(C_i) = \frac{1}{|V|} \sum_{c \in C_i} \sum_{v \in V} ndp(v, c) \quad (3)$$

The *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 $M(C_i)$.*

- *Controller failures*

To minimize the impact of such failures, the latency-based metric should consider both the distance to the (primary) controller and the distance to other (backup) controllers. For a total number of *k* controllers, the failures can be modeled [16], by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 of up to k - 1. The *Worst_case_latency_cf* will be:

$$L_{wc-cf} = \max_{v \in V} \max_{C_i \in C} \min_{c \in C_i} d(v,c) \quad (4)$$

The optimization algorithm should find a placement which *minimizes the expression* (4).

Note that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders. To minimize (4), the controllers tend to be placed in the center of the network, such that in a worst case, a single controller can take over all control. However, the scenario supposed by the expression (4) is very pessimistic; a large network could 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 in [18]. The conclusion is that an optimization trade-off should be found, for the failure-free or failure cases. A multi-criteria approach can provide the solution.

- *Nodes/links failures*

For such cases, the objective could be to find a controller placement that 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 [16]). 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 longer useful.

For a *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 an 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). In equivalent words, the algorithm counts the nodes that have no more connectivity to any controller.

$$Nlf(C_i) = \max Nlf_k(C_i) \quad (5)$$

The optimization algorithm should find a placement *to minimize* (5), where *k should* cover all scenarios of F. It is expected that increasing the number of controllers, will decrease the *Nlf* value. However, the optimum solution based on the metric (5) could be very different from those provided by the algorithms using the latency-based metrics.

- *Load balancing for controllers*

It is desired a good balance of the node-to-controller distribution. A metric $Ib(C_i)$ will measure the degree of imbalance 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:

$$Ib(C_i) = \max_{s \in S} \{ \max_{c \in C_i} n_c^s - \min_{c \in C_i} n_c^s \} \quad (6)$$

where n_c^s is the number of forwarder nodes assigned to a controller c. Equation (4) takes into account that in case of failures, the forwarders can be reassigned to other controllers and therefore, the load of those controllers will increase. An optimization algorithm should find that *placement which minimizes the expression (4).*

IV. Multi-criteria Optimization Algorithms

SDN controllers' placement and/or selection may involve several particular metrics (as summarized in Section III). If optimization algorithms for particular metrics are applied, then one can obtain different non-convergent solutions. Actually the CPP and CSP problems have naturally multi-criteria characteristics; therefore, MCDA is a good way to achieve a convenient trade-off solution.

This paper uses the same variant of MCDA implementation as in [9], i.e., the *reference level (RL) decision algorithm* [10] as a general way to optimize the controller placement, and controller selection, for an arbitrary number metrics. The MCDA-RL selects the optimal solution based on normalized values of different criteria (metrics).

The MCDA considers *m* objectives functions (whose values, assumed to be positive should be minimized). A solution of the problem is represented as a point in a space R^m of objectives; the decision parameters/variables are: v_i, i = 1, ..m, with $\forall i$, $v_i \geq 0$; so, the image of a candidate solution is $Sl_s=(v_{s1},v_{s2}, ..,v_{sm})$, represented as a point in R^m. The number of candidate solutions is S. 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 function and conforming a particular metric.

The basic MCDA-RL [10] defines two reference parameters: r_i=*reservation level*=the upper limit, not allowed to be crossed by the actual decision variable v_i of a solution; a_i=*aspiration level*=the lower bound beyond which the decision variables (and therefore, the associate solutions) are seen as similar (i.e., any solution can be seen as "good"-from the point of view of this variable). Applying these for each decision variable v_i, one can define two values named r_i and a_i, i= 1, ..m, by computing among all solutions s = 1, 2, ..S:

$$\begin{aligned} r_i &= \max [v_{is}],\ s = 1, 2, ..S \\ a_i &= \min [v_{is}],\ s = 1, 2, ..S \end{aligned} \quad (7)$$

An important modification is proposed in [16], aiming to make the algorithm agnostic versus different nature of criteria. The absolute value v_i of any decision variable is *replaced* with *distance from it to the reservation level*: r_i-v_i; (so, increasing v_i will decrease the distance); *normalization* is also introduced, in order *to get non-dimensional values, which can be numerically compared* despite their different nature. For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s, i \tag{8}$$

The factor $1/(r_i-a_i)$ - plays also the role of a weight. A variable for which the possible dispersion of values is high (max – min has a high value in formula (6)) will have lower weight and so, greater chances to be considered in determination of the minimum in the next relation (7). On the other side, if the values *min, max* are rather close to each other, then any solution could be enough "good", w.r.t. that respective decision variable.

The basic MCDA-RL algorithm steps are (see also [13]):
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}':

$$\min_s = \min\{v_{si}'\};\ i=1...m \tag{9}$$

Step 2. Select the best solution:

$$v_{opt} = \max\ \{\ \min_s\ \},\ s=1,..S \tag{10}$$

Formula (7) selects for each candidate solution s, the worst case, i.e., the closest solution to the reservation level (after searching among all decision variables). Then the formula (8) selects among the solutions, the best one, i.e., that one having the highest value of the normalized parameter. One can also finally, select more than one solution (quasi-optimum solutions in a given range). The network provider might want to apply different policies when deciding the controller placement; so, some decision variables could be more important than others. A simple modification of the algorithm can support a variety of provider policies. The new normalized decision variables will be:

$$v_{si}' = w_i(r_i - v_{si})/(r_i - a_i) \tag{11}$$

where $w_i \in (0,1]$ is a weight (priority), depending on policy considerations. Its value can significantly influence the final selection. A lower value of w_i represents actually a higher priority of that parameter in the selection process.

V. MCDA-BASED IMPLEMENTATION FOR SDN CONTROLLER PLACEMENT

A proof of concept simulation program (written in *Python* language [1] [13]) has been constructed by the authors, to validate the MCDA–RL based CPP problem and allocation of forwarders to controllers. The program has been extended in this study with reliability-related evaluation features. The simulation program uses the standard libraries and additionally the *NetworkX* and *matplotlib,* in order to create and manipulate the network graphs.

The simplifying assumptions (they could be also seen as limitations) of the model studied here, are: the network architecture is flat, i.e., no disjoint regions are defined; the network graph is undirected; any network node can be a forwarder but also can collocate a controller; when computing paths or distances, the metrics are additive; the number of controllers is predefined; the data traffic aspects and signaling interactions are not considered; the dynamic variation of the traffic in the data plane is not considered.

A. The MCDA basic model

The basic model considered in this paper, to solve the CPP and CSP problems has two working modes:

a. static mode: the input data are: network graph (overlay or physical), link costs/capacities, shortest path distances between nodes (e.g., computed with Dijkstra algorithm based on additive metric), desired number of controllers, the criteria (decision variables –these could be anyone, among those of Section III,l or others) for MCDA, and weights assigned to the decision variables).

Two working phases are defined:

(1) *Phase 1*:

1.1. *Compute all controller placements* C_1, C_2, …. (i.e., the set of candidate solutions). The number of placements is C_n^k (n= total number of network nodes; k= number of controllers).

1.3. Compute the *values of the normalized metrics for each possible controller placement (i.e., future MCDA candidate solution)*, by using specialized algorithms and metrics like those defined in Section III.

The Phase 1 phase has as outputs the set of candidate solutions (i.e., placement instances) and their associated values to fill the entries of the matrix M defined in Section IV. The Phase 1 computation could be time consuming; it depends on network size, but also on the number of criteria selected and the complexity to compute the metrics like in Section III., Such computations could be performed off-line [5]. For instance, in a real network, a master SDN controller having all these information could perform these computations. However, in a network exposing high dynamicity computing the Phase 1 in real time is a challenging issue.

(2) *Phase 2*: MCDA-RL: define r_i and a_i, for each decision variable; eliminate those candidates having parameter values out of range defined by r_i; assign – if wanted – convenient weights w_i for different decision variables; compute the normalized variables (formula (8)); run the MCDA Step 0, 1 and 2 of the (formulas (9) and (10)).

The Phase 2 provides the CPP solution.

The pseudocode of basic MCDA-based optimization processing is high level presented below:

```
Start
```

```
  //Initialize
    {Parse the arguments;
      Define the decision variables vi, i=1,..m, ∀i,
          vi ≥ 0;}
  //Build candidate solutions
       {Build the weighted graphs;
        Build candidate solutions;//S= the set of candidates
       }
  {Compute all shortest paths between pairs of
       network   nodes; // Dijkstra algorithm}
  {Normalize the decision variables; //(formula (8)
   Compute the matrix M{vsi'}, s=1…S, i=1…m;
   Compute for each candidate s, the min. among all  its normalized variables vsi';/formula (9)
   }
   Select the best solution; //formula (10)
Stop
```

b. dynamic mode : the semantic of the word *dynamic* here is the fact that some parameters are randomly generated i.e, not predefined. The initial input information is the total number of network nodes (not the complete graph) and desired number of controllers. The graph (which could be full-mesh or not) and costs of the links are randomly generated by the program.

B. Resilience-capable models

As shown in Section III, more realistic scenarios consider the possible occurrence of controller and/or network failures events. It is desired a resilient system i.e., able to recover (as much as possible) after failure events. The optimization process aims now to proactively find trade-off solutions to provide still a convenient behavior of the overall system in failure cases.

- *Backup controllers*

A simple static solution for assignment/mapping of the forwarders (this is CSP problem) to primary and backup controllers is presented below. For a given placement of the controllers, let it be C_p, the identities of nodes playing the role of controllers are known. The simplest assignment/mapping of forwarders to controllers is based on the shortest path (metric is average estimated latency forwarder-controller) to a controller. So, an algorithm will compute, for a given placement C_p, the distances from each F_i to each controller CT_1, CT_2, …CT_k and select the closest controller, let it be CT_m, as primary controller for F_i.

How to define the backup controllers? A natural solution (supposing that the total number of controllers is still k) will be to allow a forwarder to migrate from a failed primary controller to another backup/secondary controller, selected from the same set. This backup controller can be determined by the above algorithm, as the second one in the ordered list (using the shortest distance as criterion). This assignment should be performed for every possible placement C_i. If CPP optimization and forwarders-to-controllers assignment is wanted for the backup controllers, then it is necessary to add a new criterion (decision variable- e.g., similar to the average distance given by the formula (1)) to the MCDA algorithm, with a metric similar to that of formula (1). The reason is that for primary controller placement and forwarder assignment, one can find C_i as the best solution while for and backup controller placement and assignment other different C_j could be the best. Therefore, the MCDA can provide the best trade-off.

An auxiliary algorithm is used to compute a simple metric (average distance to a backup controller) to be added to MCDA. We introduce a novel decision variable *dist_backup* and perform the following computation (for each possible controller placement Ci containing the controllers CT_1, CT_2, ….CT_k):

```
For each forwarder Fi, i=1..N
 Do
   Dist_backup = 0;
   Compute dist. from Fi to any CTj, j=1..k;
   Dist_backup=Dist_backup + second_shortest_cost;
  Od
   Dist_backup_avg = Dist_backup/N;
```

This `Dist_backup_avg` can be added as a new decision variable to MCDA (maybe with appropriate wight) Therefore, the optimization will select a solution which considers also the backup controller placement and assignment of forwarder nodes as a factors influencing the final solution selection.

A simple example (Figure 1) will show the need of the additional MCDA criterion for the backup controllers. The example network is represented by an undirected graph, where the metric indicated on the edges can be the average latency between nodes (vertices).

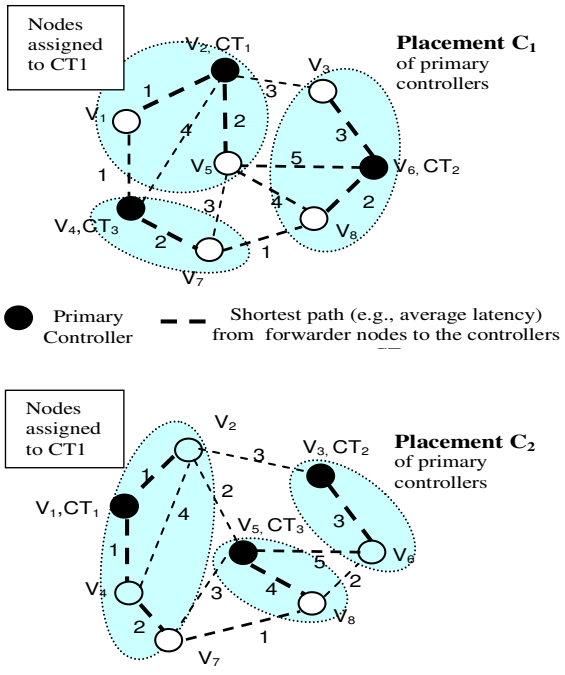


Figure 1. Simple example of two instances of primary controller placements and forwarders assignment

The network nodes are denoted with V_i, in order to emphasize that a given node can play the role of a forwarder but also a controller can be installed there. The edges costs correspond to an additive metric, i.e the average estimated communication delay between two nodes. The edges can represent real links or overlay ones (this aspect is irrelevant for the purposes of this study). It is assumed that three controllers exist, CT_1, CT_2, CT_3. We consider two placements of the *primary* controller placement C_1 and C_2. If one uses the optimization criterion given by the formula (1), then the assignment of forwarders to controllers are determined by the *shortest path* of each node to CT_1, CT_2, CT_3. The list of ordered distances can also provide the identity of the backup controller for each node V_i. Table I clarifies the assignment of the primary and backup controllers of the placement C_1. A similar table is valid for C_2, etc.

TABLE I. EXAMPLE: DISTANCES FROM NODES TO CONTROLLERS AND BACKUP CONTROLLERS DETERMINATION (PLACEMENT C_1)

				Selection	
Controller/ Node	**CT_1**	**CT_2**	**CT_3**	**Primary**	**Backup**
V_1	**1**	6	1	CT_1	CT_3
V_2	**0**	6	2	CT_1	CT_3
V_3	3	**3**	5	CT_2	CT_1
V_4	2	4	**0**	CT_3	CT_1
V_5	**2**	5	4	CT_1	CT_3
V_6	6	**0**	4	CT_2	CT_3
V_7	4	3	**2**	CT_3	CT_2
V_8	5	**2**	3	CT_2	CT_3

Considering the values of Table I the best assignment of forwarders to controllers for C_1 placement, is:
Primary controllers:
CT_1: {V_3, V_4}, CT_2: {V_3, V_6, V_8 }, CT_3: { V_4, V_7}
Backup controllers:
CT_1: {V_3, V_4}, CT_2: {V_7}, CT_3: { V_1, V_2, V_5, V_6, V_8 }
Analyzing the results two conclusions can be drawn:

- the assignment of the fowarders to primary controllers and respectively backup, can be very different
- the balance between solutions can be also very different; one can see the unbalance of the backup controller assignment.

Simple computations show that the average values of distances for the primary and respectively backup controllers are 1.25 and 2.75.

For another placement instance, i.e., C_2 (see Figure 1) one gets:
Primary controllers:
CT_1: {V_1, V_2, V_4, V_7}, CT_2: {V_3, V_6}, CT_3: { V_5, V_8}
Backup controllers:
CT_1: {V_3, V_5, V_8}, CT_2: {Ø}, CT_3: { V_1, V_2, V_4, V_6, V_7 }

The average values of distances for the primary and respectively backup controllers are 1.5 and 3.5. So one can say that C_1 placement is a better solution.

Even such simple examples prove the real need and usefulness of multi-criteria optimization, where resilience-oriented metrics can be added.

- *Load balancing for controllers*

As shown in Section III, a good balance of the node-to-controller distribution is desired as a proactive procedure to minimize the chance of future controller overload and to provide fairness between controllers. This paragraph will propose a simple load balancing solution for controllers. The solution is static, i.e., it will try to assign to different controllers, approximately, the same number of forwarders to be controlled. Note that such a solution will produce enough good results during the run-time, only if the data plane traffic distribution between the forwarders is rather uniform.

If the total number of nodes is N and the number of controllers is k, then the average number of nodes allocated to a controller is N/k. A simple new metric can be added to the set of MCDA criteria. This decision variable *D_avg* will measure the deviation of the actual number of nodes allocated to a controller CT_i, i.e., n_i, from the average value N/k, and averaging this for all controllers.

$$D\_avg = (1/N) \sum_{i=1\ldots k} |(n_i - N/k)| \quad (12)$$

If wanted, this variable can get an appropriate weight in the multi-criteria optimization process. If the example of the previous sub-section on backup controller problem is considered, then one can learn that solutions found there (based on latency criteria) could expose significant unbalance between controllers.

- *Nodes and link failures*

Nodes and link failures could appear in the network. Evaluation of effects of such events could be taken into account by adding new decision appropriate parameters in the set of MCDA input multi-criteria. Here, we adopted a different approach in comparison with the metric presented in Section III [7]. Given that most important metrics are forwarder-controller latency, inter-controller latency, load balancing of the controllers, optimization of the placement of the primary and backup controllers, the MCDA has been first run to produce controllers' placement optimization based on these important parameters. Then the simulation program allows some events to happen (e.g., nodes or link failures). The MCDA has been run again and produce a new placement after removing from the graph the failured entities. Finally, the placement produced in the updated conditions can be compared with the initial one, to evaluate if significant changes appeared. In such a way one can evaluate the robustness of the initial placement, and decide if that can be preserved or must be changed.

Two input parameters have been defined in the model:
nf- number of nodes supposed to fail
ef – number of links supposed to fail.

The specific nodes and links which will fail will be selected as to to simulate the "worst case", i.e., those nodes having the lowest cost of the adjacent links and, respectively those links having the least costs. If after second run of the MCDA, the initial placement of the controllers does not change, this means that initial placement has enough good robustness properties. Of course, this result will depend on selection of nf and ef values, for a given N nodes of the graph.

C. Simulation program for controller placement and selection optimization

The user interface of the simulation program (having resilience features included) is presented in Figure 2.

```
stefan@mint ~/Desktop/simulator_mcda $ python mcda.py -h
usage: mcda.py [-h] [-a [A]] [-w [W]] [-i [I]] [-b [B]] [-l [L]] [--dynamic] [-n N] [-c C] [-nf NF] [-ef
EF] [--debug]

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].
  -b [B]      Average latency - failure scenario. Expects a weight (priority) in interval (0, 1].
  -l [L]      Controller load-balancing. 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
  -nf NF      Number of nodes that fail. Valid only in dynamic mode. Allowed  values: 1.. N-C.
  -ef EF      Number of edges that fail. Valid only in dynamic mode. Allowed  values: 1 ..N-C.
  --debug     Prints some computing results for debugging purposes.
```

Figure 2. The interface of the MCDA CPP simulation program

The decision parameters considered have been: *average* and *worst* latency between a forwarder and controller, *inter-controller* latency and *load balancing* related parameter. The program can be run in static or dynamic mode, with any number and set of criteria among those presented in the interface. The program is flexible in the sense that the set of decision weighted parameters (having appropriate metrics) can be enriched at will; the only needed modification is the number of columns of the matrix M.

Several numerical examples and results of the basic CPP solutions have been already presented in the work [13]. The current version of the implementation added reliability feature presented in Section IV.B.

The pseudo-code of the simulation program for dynamic mode is presented below, in high level view.

```
Start
   Generate the random graph;
   Generate all controlers' placements;
   Run MCDA;
   If link_failures are specified as a running
option then eliminate from the graph a number of
ef links having the minimum costs;
   If node_failures are specified as a running
option then eliminate from the graph a number of
nf nodes;
   If failures_are produced
      then {generate modified graph; Run MCDA;}
   Display the graphs;
Stop
```

D. Dynamic controller selection

In a dynamic network context, the controller selection (CSP) can be performed in a dynamic way. The multi-criteria algorithm can be as well applied in such cases. We consider here only the situations in which controller/node/link – failures occur.

In the static approach the backup controllers are predefined; their placement is selected by the optimization algorithm. For a real network, the algorithm can be run offline in a management center (in a hierarchical organization of the control plane, this could be a master SDN controller). This center is supposed to know all information in order to run MCDA-RL algorithm. The aspects related of collecting this information at the master SDN controller constitute a separate problem, which is not studied in this paper.

If a running forwarder looses its connectivity with its primary controller, it can act in two ways; a. try to connect to a known backup controller; b. select among several available controllers by running a MCDA algorithm. The input information for MCDA (decision criteria) could be:

- identities/addresses of a set of SDN controllers;
- degree of load for those controllers (e.g., periodically communicated, by a traffic monitoring system (having its central intelligence in the master SDN controller) to the forwarder
- local information observed by the forwarder, like connectivity to different nodes/controllers, etc. So, the forwarder can select based on MCDA-RL a novel controller.

VI. EXPERIMENTAL RESULTS

This section will shortly present some simple but relevant examples of results, in order to prove the validity of approach. The experiments are mainly oriented to test the resiliency.

- *Basic controller placement examples*

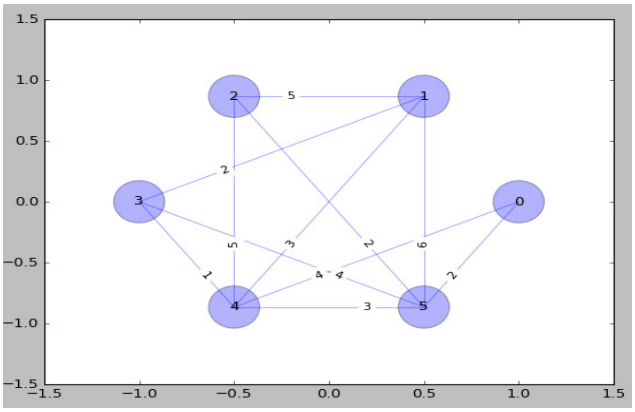


Figure 3. Static MCDA CPP optimization with individual criteria in MCDA

The objective of the first example is to show that applying a single optimization criteria the solutions can be very different. Figure 3 shows an example of optimization for a network statically defined, having N= 6 nodes and c= 2 controllers. Four controller placements have been considered:C_0: nodes [4, 5]; C_1: nodes [2, 4]; C_2: nodes [2, 5]; C_3: nodes [3, 5].

The result placements (examples) for different individual criteria are listed below:

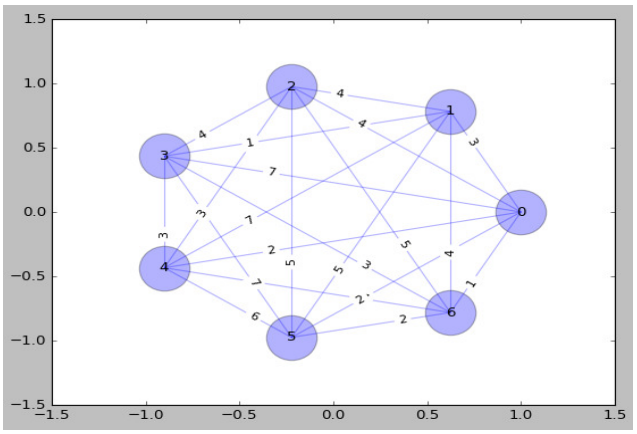
```
stephan@mint$ python mcda.py -a //average latency
to the primary controllers
Optimum Ci placement is Ci=3;
CT0 is placed in node 3
CT1 is placed in node 5
CT0 nodes {1,3,4}
CT1 nodes {0,2,5}
```

The computed latencies for the four placements are : C_0: 1.33; C_1: 1,66; C_2: 2,33; C_3: 1.13. One can see that C_3 is the best.

```
stephan@mint$ python mcda.py -b //average latency
to the backup controllers
Optimum Ci placement is Ci=0
CT0 is placed in node 4
CT1 is placed in node 5
CT0 nodes {1,3,4}
CT1 nodes {0,2,5}

stephan@mint$ python mcda.py -w //max latency to
the backup controllers
Optimum Ci placement is Ci=3
CT0 is placed in node 3
CT1 is placed in node 5
CT0 nodes {1,3,4}
CT1 nodes {0,2,5}

stephan@mint$ python mcda.py -i //inter-controller
latency
Optimum Ci placement is Ci=2
CT0 is placed in node 2
CT1 is placed in node 5
CT0 nodes {1,2}
CT1 nodes {0,3,4,5}
```



. Figure 4. Basic MCDA CPP optimization with dynamically generated network graph

The variety of the above results obtained for single criterion shows clearly the necessity of a multi-criteria optimization.

The second example shows a multi-criteria scenario. Figure 4 shows a graph dynamically generated with N=7 nodes and k=2 controllers. The optimization criteria have been *average latency*, *worst latency* and *inter-controller latency*, with equal weights $d_1=d_2=d_3=1$. The best placement selected is C2 , having the controllers placed in the nodes 0 and 3. The allocation of forwarders to controller can be selected based on shortest path principle. The command to run program and the main results are listed below.

```
stephan@mint$ python mcda.py -a 1 -w 1 -i 1 -
dynamic -n 7 -c 2
Optimum Ci placement is Ci=2
Controller is placed in node 0
Controller is placed in node 3
```

- *Load balancing for controllers*

Figure 5 shows an example in which the network graph has been dynamically generated with N=6 nodes and k= 2 controllers. The decision criteria have been *inter-controller latency* (weight = 1) and *balancing criterion* (weight = 0.5, i.e., having twice higher priority). The MCDA program has been run with parameters :

```
   stefan@mint$ python mcda.py -i 1 -l 0.5 --
dynamic -n 6 -c 2
```

The results obtained are: controllers CT_0 and CT_3 placed in the nodes 0 and 3. The allocation of forwarders to controllers are :

```
   Controller 0 has allocated node(s): 0, 2, 4.
   Controller 3 has allocated node(s): 1, 3, 5.
```

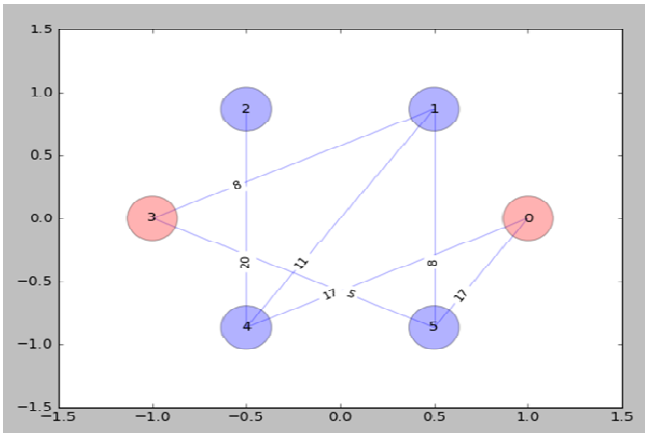


Figure 5. Example of a balanced allocation of the forwarders to controllers (after MCDA run)

Note that in this solution the inter-controller latency is taken into consideration, but the final value is not minimum; however, the allocation of the forwarders to controllers is balanced (3 forwarders per each controller). The reason is that load balancing criterion has been assigned a higher priority versus the inter-controller-latency.

- *Links and node failures*

To experiment such scenarios the simulator should be launched in *dynamic mode* and the number of links/nodes which will be in failure should be also specified. One can check if the placement selected is resilient to failures. For instance, if the unique parameter considered in MCDA would be the *average latency* of the *forwarders* to *backup* controllers, then one would expect that the resulting placement could be enough resilient to a low number of nodes and/or link failure events. Figure 6 shows such an example, by presenting the graphs resulted after running the program with the command:

```
python mcda.py -b --dynamic -n 8 -c 3 -ef 2
```

In this example, the network has N=8 nodes and c= 3 controllers; the number of failure links ef=2. This first placement (Figure 6a) has the controllers installed in nodes 3,4,5. The program is run again after some links failure (1-6, 3-7). Still the controller placement (i.e., after running again the MCDA on the reduced graph) is the same (Figure 6 – right), i.e., in the nodes 3,4,5.

Now we consider an experiment in which the criterion of the first run of the MCDA is to minimize the *average latency* between the *forwarders* and *primary* controllers (parameter introduced with weight = 1). The optimum placement of the controllers (with N=8, c=3), after first run of the MCDA, is in nodes 0, 2, 6. (Figure 7, left). Then two link failures are simulated (i.e., links 5-6 and 0-1 will be out of order). The command for such a run is:

```
python mcda.py -a --dynamic -n 8 -c 3 -ef 2
```

The optimum placement of the controllers in the new context (failure links) has been changed in nodes 3,5,6 (Figure 7b). So, one can conclude that the first placement is less resilient to link failures.

The lesson learned from such experiments is that there is no absolute unique optimum solution of such problems, to satisfy all requirements. Depending on the particular context of the SDN-controlled network and some network owner policies, different placement solutions can be found as more appropriate, to satisfy in a convenient way several criteria.

These examples illustrate the power of the MCDA algorithm where various sets of criteria and different priorities (driven by policies) can be considered.

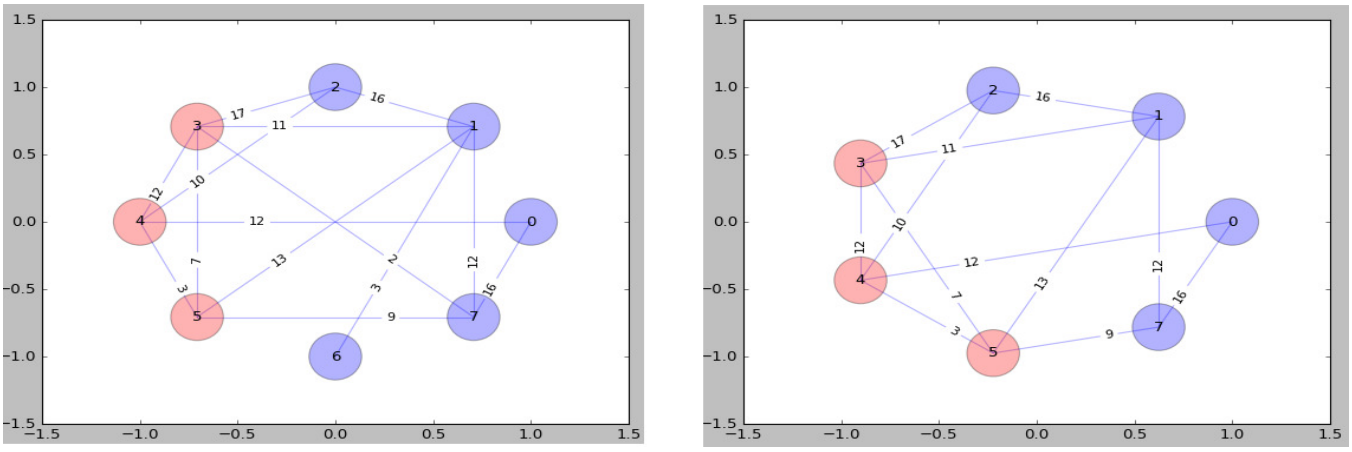


Figure 6. Example of controller placement resilient to link failures a.Left: placement before link failures; b.Right: placement after some links failures.

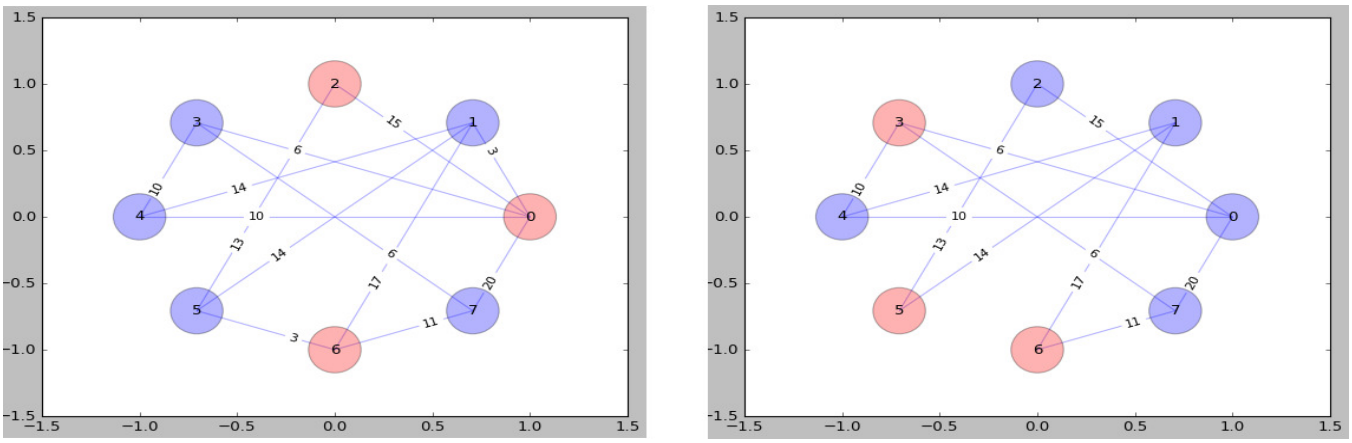


Figure 7. Example of placement non-resilient to link failures
a.Left: controller placement before link failures; b. Right: controller placement after some links failures (5-6, 0-1).

VII. CONCLUSIONS AND FUTURE WORK

This paper extended the study [1], on using multi-criteria decision algorithms (MCDA) to optimally place the controllers in large SDN, based networks, while aiming to achieve good resiliency properties of the system. It is illustrated the main MCDA advantage, i.e., that it can produce a tradeoff (optimum) result, while considering several weighted criteria, part of them even being partially contradictory.

This study provides (in comparison to [1]) more comprehensive discussion and analysis of resiliency-oriented properties of a SDN network with distributed control plane. Simple but relevant examples have been added, to show that actually no unique solution exists for controller placement to be optimal with respect to all criteria envisaged. Therefore, in practice, the weights of the decision parameters introduced in MCDA should be cleverly adopted, to meet the prioritized list of the network provider requirements.

This study has shown that actually the MCDA – based optimization can be performed in a flexible way:

- introducing in MCDA all decision parameters, with appropriate weights in order to achieve a trade-off solution after a single MCDA run;
- using iteratively several rounds (see Section VI), i.e., introducing first the most important parameters and run MCDA; then modify the topology/conditions and check if the first controller placement is still good enough in these new conditions; if not, then add parameters to MCDA and run again the algorithm.

The paper added several additional experimental results in Section VI. The forwarder-controller mapping optimization and backup controller selection have been also considered.

Future work will be still necessary for CPP and CSP problems. Experiments on large networks [22][23] could better validate the optimization solutions in a more realistic environment. Another important aspect can be the dynamic of the overall system during run-time, when the traffic amount inside different regions of the the data plane (i.e., between different forwarders) might have significant variations. This can lead to overload of some SDN controllers, especially if reactive-mode of the control plane is applied in those networks and given the limited controller processing capacity. This problem could be solved in two ways: a. moving some controllers (so, the placement will be modified) to the overloaded regions to better serve the forwarder requests for flow table configuration); b. dynamically migrate some switches/forwarders between the controllers, in order to better balance the controllers' load. For instance, in recent studies [20][21], the dynamic switch migration is optimized based on input information periodically provided by a monitoring system. However, these studies do not consider multi-criteria approach, but only the traffic load of the network data plane and impact on controller tasks. Here, combining MCDA with such traffic-based algorithms could provide better results.

REFERENCES

[1] E. Borcoci and S. Ghita, "Reliability-aware Optimization of the Controller Placement and Selection in SDN Large Area Networks", The Thirteenth International Conference on Systems and Networks Communications, ICSNC, Nice, 2018, https://www.iaria.org/conferences2018/ICSNC18.html, [retrieved: 6, 2019].

[2] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On Scalability of Software-Defined Networking", IEEE Comm. Magazine, pp. 136-141, February 2013.

[3] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010, https://pdfs.semanticscholar.org/f7bd/dc08b9d9e2993b363972b89e08e67dd8518b.pdf, [retrieved: 5, 2018].

[4] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, et al., "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010, https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Koponen.pdf, [retrieved: 5, 2018].

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualisation: Challenges and Opportunities for

Innovations", IEEE Communications Magazine, pp. 90-97, February 2015.

[6] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, pp. 7–12, 2012, https://dl.acm.org/citation.cfm?id=2342444, [retrieved: 5, 2019].

[7] K. Sood and Y. Xiang, "The controller placement problem or the controller selection problem?", Journal of Communications and Information Networks, Vol.2, No.3, pp.1-9, Sept.2017.

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

[9] E. Borcoci, T. Ambarus, and M. Vochin, „Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes," The 15th International Conference on Networks, ICN 2016, Lisbon, http://www.iaria.org/conferences2016/ICN16.html, [retrieved: 5, 2019].

[10] 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.

[11] S. Yoon, Z. Khalib1, N. Yaakob, and A. Amir, "Controller Placement Algorithms in Software Defined Network - A Review of Trends and Challenges", MATEC Web of Conferences ICEESI 2017 140, 01014 DOI:10.1051/matecconf/201714001014, 2017.

[12] G.Wang, Y.Zhao, J.Huang, and W.Wang, "The Controller Placement Problem in Software Defined Networking: A Survey", IEEE Network, pp. 21- 27, September/October 2017.

[13] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN", International Journal of Network Management, March 2018, pp 1-25, https://doi.org/10.1002/nem.2018.

[14] A.Kumari and A.S.Sairam, "A Survey of Controller Placement Problem in Software Defined Networks", arXiv:1905.04649v1 [cs.NI] 12 May 2019.

[15] H. Yan-nan, W. Wen-dong, G. Xiang-yang, Q. Xi-rong, and C. Shi-duan, "On the placement of controllers in software-defined networks", ELSEVIER, Science Direct, vol. 19, Suppl.2, pp. 92–97, October 2012, , http://www.sciencedirect.com/science/article/pii/S10058851160438X, [retrieved: 1, 2018]..

[16] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," Proceedings of the ITC, Shanghai, China, 2013, https://ieeexplore.ieee.org/document/6662939/, [retrieved: 1, 2019].

[17] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability aware controller placement for software-defined networks," in Proc. IM. IEEE, pp. 672–675, 2013, https://ieeexplore.ieee.org/document/6573050/, [retrieved: 1, 2019].

[18] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, and M. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability", IEEE Global Comm. Conference (GLOBECOM); 12/2014, https://ieeexplore.ieee.org/document/7037087/, [retrieved: 4, 2018].

[19] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks" in GLOBECOM 2011, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.691.795&rep=rep1&type=pdf, [retrieved: 5, 2019]. .

[20] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, et al., "Balcon: A distributed elastic SDN control via efficient switch migration", in Proc. IEEE Int. Conf. Cloud Eng. (IC2E), April 2017, pp. 40–50.

[21] Y.Xu, M.Cello, I-Chih Wang, A. Walid, G. Wilfong, et al., "Dynamic Switch Migration in Distributed Software-Defined Networks to Achieve Controller Load Balance", IEEE Journal on Selected Areas in Communications , Vol. 37, No. 3, March 2019, pp. 515-528.

[22] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/, [retrieved: 4, 2018].

[23] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011, pp.1765-1475.

[24] F. Al-Tam and N.Correia, "On Load Balancing via Switch Migration in Software-Defined Networking", IEEE Access, DOI 10.1109/ACCESS.2019.2929651, pp.1-13, July 2019.

Misuse Detection in Dynamic Spectrum Sharing Wireless Networks Across Multiple Channels

Debarun Das
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: ded59@pitt.edu

Taieb Znati
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: znati@pitt.edu

Martin Weiss
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: mbw@pitt.edu

Pedro Bustamante
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: pjb63@pitt.edu

Marcela M. Gomez
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: mmg62@pitt.edu

J. Stephanie Rose
School of Computing and Information
University of Pittsburgh
Pittsburgh, USA
Email: jsr67@pitt.edu

Abstract - **We propose a spectrum enforcement framework across multiple channels by mobile, crowdsourced agents (also called volunteers), who work in collaboration with a trustworthy infrastructure. The success of spectrum sharing relies on the automated enforcement of spectrum policies. The primary challenge addressed here is to ensure *efficient ex post* spectrum enforcement. In order to achieve this, we focus on attaining maximum coverage of the area of enforcement and of all channels, and on ensuring reliable and accurate detection of spectrum violation. Maximum coverage of the given area of enforcement is ensured by proposing to divide it into smaller regions using the Lloyd's algorithm and solving the enforcement problem by a divide and conquer mechanism over the entire area. We determine the qualification of volunteers based on their likelihood of being in a region, and on their trustworthiness. We define algorithms to select qualified volunteers for every region in an online manner such that every channel is efficiently covered. The enforcement framework is simulated in CSIM19 (C++ version) and extensive analysis of the performance of the proposed methodologies is performed.**

Keywords- volunteer; sentinel; ex post spectrum enforcement; crowdsourced spectrum enforcement; volunteer selection; channel assignment; mobility.

I. INTRODUCTION

With the exponential increase in use of wireless services, the demand for additional spectrum is steadily on the rise. In order to address this potential spectrum scarcity problem, the Federal Communications Commission (FCC) proposed Dynamic Spectrum Access (DSA), wherein licensed frequency bands when idle, are utilized by unlicensed users. In April 2015, the FCC adopted a three-tiered spectrum sharing infrastructure that is administered and enforced by Spectrum Access System (SAS). This architecture consists of Incumbents in tier 1, Priority Access Licensed (PAL) devices in tier 2 and General Authorized Access (GAA) devices in tier 3. Incumbents, in general, include military radars, fixed satellite service Earth stations and several of the Wireless Broadband Services (3650 – 3700 MHz) [2]. The SAS ensures that the spectrum is always available to the incumbent users when and where needed. The next level of access is provided to the users who buy PAL for a given location and period of time (usually for a three-year term). The remaining spectrum can then be used by devices having GAA. These devices have no protection from interference. They must, however, protect incumbents and PALs, while accessing spectrum [2].

As spectrum sharing becomes more intense and more granular with more stakeholders, we can expect an increasing number of potentially enforceable events. Thus, the success of spectrum sharing systems is dependent on our ability to automate their enforcement. The three key aspects of any enforcement regime are — the timing of enforcement action, the form of enforcement sanction and whether the enforcement action is private or public [3]. This paper focuses on detection of spectrum misuse. Thus, the key aspect of enforcement action for our consideration, is the timing of enforcement. Timing of an enforcement can be either *ex ante* (before a potentially "harmful" action has occurred) or *ex post* (after a potentially "harmful" action has occurred, but potentially before or after an actual "harm" has been done) [4]. The *ex ante* and *ex post* enforcement effects are inextricably linked. For example, if the *ex ante* rules and processes are sufficiently strong then *ex post* harms may be prevented before they occur. Also, certain types of *ex ante* rules may be easier to monitor and hence lower the cost of enforcement. Even strong *ex ante* rules may require *ex post* enforcement; for example, licensing approval for equipment is usually based on a prototype or pre-production unit, but compliance of production units may require some kind of policing. Till date, more significance has been given on automating *ex ante* enforcement of usage rights. As an example, the TV White Spaces database systems essentially work by preventing users with subordinate rights from using spectrum when and where other users with superior rights are

operating [5]. This concept has been extended in the new Citizens Broadband Radio Service (CBRS) to a SAS that is designed to distinguish the three classes of user types discussed previously [2].

We observe that both SAS and CBRS have well-developed mechanisms to avoid interference but provide no support for addressing interference when it occurs. As we consider *ex post* enforcement approaches, the need to detect enforceable events, gather information about these events and adjudicate claims based on rules and evidence becomes important. In this paper, we focus on designing an efficient framework for the detection of an interference event that is caused by a malicious user. The primary challenge is to ensure efficient *ex post* spectrum enforcement. In order to address this challenge, this paper proposes an enforcement framework that aims to achieve a) maximum coverage of the entire area of enforcement, b) maximum coverage of all the channels in a region c) an accurate, reliable and feasible detection of an event of violation, d) use of an effective method for hiring and deploying detecting agents. We leverage crowdsourced spectrum enforcement because it is more cost-effective and has the potential for higher accuracy of detection and localization of spectrum access violation when compared to static enforcement [13][28]. By employing a hybrid infrastructure of crowdsourced and trusted, dedicated resources, we aim to ensure "optimal" detection of spectrum access violation in Dynamic Spectrum Sharing Wireless networks. The major contributions of this paper are:

a) *Region Coverage*: We use a clustering algorithm to organize the area into smaller sized "regions" in order to ensure more manageable detection of violation. The enforcement problem can then be solved by a divide and conquer mechanism over all the regions.
b) *Channel Coverage*: We develop an algorithm to ensure efficient coverage of all channels in a region.
c) *Crowdsourced Enforcement*: We explore a mechanism to select crowdsourced agents (also called volunteers) for ensuring that a spectrum access violation is detected with high probability of accuracy and efficiency.
d) *Volunteer Selection*: We develop a framework to assess the *qualification* of a volunteer across two dimensions — location likelihood and trust, which is used to select volunteers such that an "optimal" quality of spectrum enforcement is ensured.

The paper is organized in the following manner. Section II of the paper discusses about the related works, while Section III of the paper discusses about the proposed enforcement framework. Section IV discusses about the crowdsourced monitoring methodology, with a focus on the parameters that qualify a volunteer for selection and the appropriate volunteer selection mechanism. Section V discusses about the experimental setup and the results we obtained from applying the proposed volunteer selection algorithm. Finally, we conclude the paper and discuss about future works in Section VI.

II. Related Works

Jin *et al.* [20] introduce the first crowdsourced spectrum misuse detection framework for DSA systems, where a legitimate transmitter is required to embed a spectrum permit into its physical layer signals, which can be decoded and verified by ubiquitous mobile users. Dutta and Chiang [13] discuss about crowdsourced spectrum enforcement for accurate detection and location of spectrum enforcement. However, they assume that crowdsourced spectrum access enforcers are trustworthy and do not examine the effect of distrust of enforcers. Li *et al.* [23] model the spectrum misuse problem as a combinatorial multi armed bandit problem to decide which channels to monitor, how long to monitor each channel, and the order in which channels should be monitored. However, they assume that the spectrum monitoring agent and the malicious users are always static. Salama *et al.* [22] proposed an optimal channel assignment framework for crowdsourced spectrum monitoring, where volunteers are assigned to monitor channels based on their availability patterns and are awarded with incentives in return. Several incentive-based crowdsourced spectrum sensing works have been done over the past few years. Yang *et al.* [7] studied two incentive-based crowdsourcing models, where a Stackelberg Equilibrium was computed in the platform-centric model, and a truthful auction mechanism was proposed under the user-centric model. Zhu *et al.* [14] propose an incentive-based auction mechanism to improve fairness of bids by taking into consideration the effects of malicious competition behavior and the "free-riding" phenomenon in crowdsourcing services. Lin *et al.* [6] take the Sybil attack into consideration for incentive-based crowdsourced spectrum sensing. The works [11] and [12] propose frameworks for crowdsourced spectrum sensing without violating the location privacy of mobile users. Contrary to majority of the formerly proposed spectrum monitoring approaches, which rely exclusively either on large deployment of physical monitoring infrastructure [8]-[10] or on crowdsourcing, we believe that spectrum misuse and access rights violations can be effectively prevented by using trusted infrastructure (composed of a central DSA Enforcement Infrastructure and a minimal number of mobile, wireless devices with advanced trust and authentication capabilities), augmented with an opportunistic infrastructure of wireless devices with various software and hardware capabilities. Moreover, in contrast to the usual methodologies, we explore the use of an online non-incentive-based methodology for selection of mobile volunteers based on their *qualifications* to ensure maximum coverage of enforcement area, efficient coverage of all the channels in an enforcement region and accurate detection of spectrum access violations. This work is an extension of our

previous work [1]. Contrary to our previous work, in this paper, we propose spectrum enforcement over multiple channels. We explore multiple ways to aggregate the different parameters for the calculation of qualification of a volunteer and develop an efficient algorithm for assignment of channels to the selected volunteers for monitoring. In contrary to our work in [27], we explore the effect of different parameters (trust and location likelihood) in the performance of the crowdsourced agents. Finally, in contrast to [1] and [27], we conduct more experiments and analyze the results for a more comprehensive and extensive evaluation of our system.

III. ENFORCEMENT FRAMEWORK

The main challenge in the design of a hybrid infrastructure stems from the fact that it is not easy to determine where and how the resources are to be mobilized, given the non-deterministic nature of mobile devices' *behavior*. It is equally difficult to determine how collaboration between these devices must take place to ensure swift detection and response to spectrum misuse and access rights violation. To address this, we broadly follow a crowdsourced monitoring infrastructure, supported by sentinel-based monitoring and a central DSA Enforcement Infrastructure.

A. System Model

The entire area of enforcement R is divided into smaller regions, with an Access Point AP_r, associated with every $r \epsilon R$. Authorized users, who are legitimate Secondary Users (SUs) gain access to an available channel through the local AP_r in r. On the contrary, malicious users are unauthorized transmitters who intrude on spectrum by illegitimately using spectrum frequencies in r that they have not been authorized to use by the local AP_r. Some of the authorized users volunteer to monitor a given channel for access violation, in addition to accessing the spectrum to transmit their own data. Such volunteers are mobile agents who can monitor radio access behavior within their neighborhood and detect anomalous use of spectrum. To carry out spectrum monitoring practices, volunteers incur transmit power consumption cost and bandwidth consumption cost.

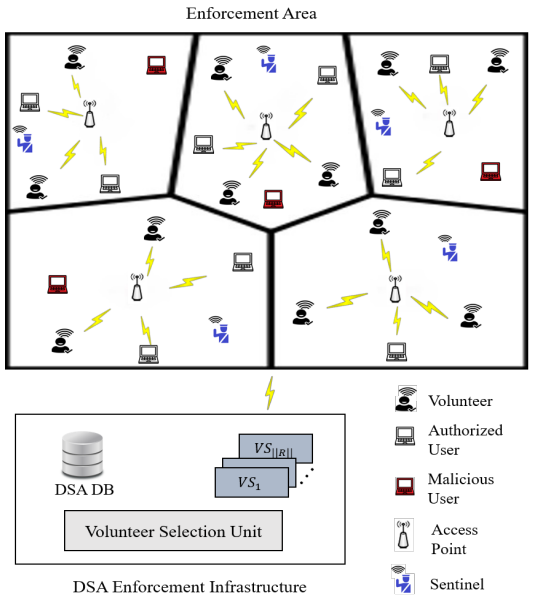


Figure 1. System Model.

As shown in Figure 1, the system model further consists of a central DSA Enforcement Infrastructure, which consists of a set of Volunteer Service units VS_r for every $r \in R$, a Volunteer Selection Unit and a DSA Database. A volunteer $v \epsilon V$ in $r \in R$ registers itself to the VS_r associated with r. A VS_r stores and updates volunteer attributes over the entire period of enforcement. The Volunteer Selection Unit uses the latest attributes of all the volunteers in a VS_r to select volunteers for monitoring a given channel in r over the next epoch of enforcement. The DSA Database maintains a channel-user occupancy list, for the entire area of enforcement R. The information contained in the DSA Database is used to identify the channels and their corresponding authorized users in R. Finally, the system model consists of a set of sentinels S' who monitor a given channel in r at random intervals to verify the detection results reported by the volunteers and to prevent selection of volunteers who have unreliable behavior.

B. Coverage of Region

To ensure maximum coverage of an area R for enforcement, we follow a divide and conquer method. We propose to divide the entire area R into smaller regions and then focus on solving the enforcement problem for a single region $r \in R$. This in turn can be used for solving the problem for the whole R. For division of R into regions, we propose the employment of the Voronoi algorithm [15]. Initially, we assume that the volunteers in V are randomly distributed over R and the access points are spread uniformly over R. For each volunteer $v \in V$, its corresponding Voronoi region r consists of every volunteer in the Euclidean plane whose distance to the local AP_r is less than or equal to its distance to any other AP_r [15]. However, the Voronoi algorithm may not produce regions that are of equal size. This is a disadvantage because it may result in some of the regions to have an undersupply of volunteers over time, which in turn may result in possible loss in detection of spectrum violation. Thus, we propose to apply a relaxation to the Voronoi algorithm, called the Lloyd's Algorithm [16], which produces uniformly sized convex regions, and thus improves the probability of a fair distribution of volunteers over all regions. The number of regions in R is equal to the number of access points in R.

IV. CROWDSOURCED SPECTRUM MONITORING

A volunteer $v \in V$ is associated with the following parameters: Serial Number of the sensing device S_v used by

v and its location $L_{v,t}$ at time t. While S_v can be used to uniquely identify a volunteer, the location $L_{v,t}$ allows the VS_r of the DSA Enforcement Infrastructure to estimate whether v will be available to monitor a given channel in r in the future.

As shown in Figure 2, we divide the total enforcement time into a set of intervals called the Monitoring Intervals, MIs. Each MI is further divided into a set of n sub-intervals called the Access Unit Intervals (AUIs). One AUI is defined as the smallest interval over which a user, intruder or legitimate, can accomplish useful work. It is used as the interference monitoring interval by the selected volunteers to determine spectrum access violation or legitimacy. A new set of volunteers is selected in region r at the end of every MI by the Volunteer Selection Unit using the data from Volunteer Service unit VS_r associated with region r. Volunteer selection in r is primarily based on twofold parameters of trust and location likelihood of a v in r.

A. Trust

The trust of a volunteer v is determined by its past behavior. The behavior of a volunteer v is chiefly determined by its accuracy in detection of spectrum access violation. At the end of every AUI i, a volunteer v reports the observed state $\phi^i_{v,r,c}$ of a channel c that it monitors in region r, over i. The state of a channel c can be either a) violated, when c is being used by a malicious transmitter b) not violated, when c is either idle, i.e., when no user, authorized or malicious, uses c or safe, i.e., when c is used by an authorized transmitter. The necessary ground truth required for calculating accuracy of interference detection by v in r is acquired from the observed state $\phi^j_{s,r,c}$ of c by a sentinel $s \in S'$ that monitors c at a random AUI j in the given MI. A sentinel s is a trustworthy agent who helps in verifying volunteer detection result and helps to identify unreliable volunteers. As shown in Figure 2, a sentinel s monitors c in r at a random interval j,which is not known to the volunteers. This helps us to calculate the behavior $b^i_{v,r,c}$ of v in r at AUI i by using (1) given below.

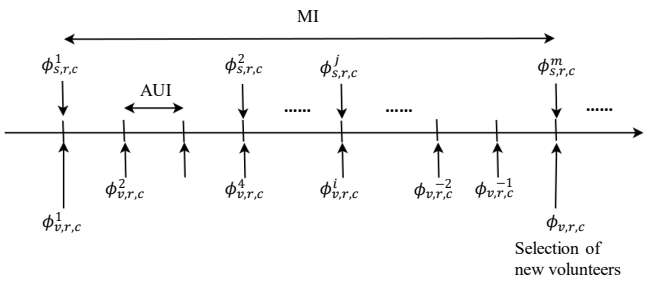


Figure 2. Observations $\phi^i_{v,r,c}$ by volunteer v after every AUI and $\phi^j_{s,r,c}$ by sentinel s after random AUIs, for the 1st MI.

$$b^i_{v,r,c} = \begin{cases} 1, & \phi^i_{v,r,c} = \phi^j_{s,r,c} \\ 0, & \phi^i_{v,r,c} \neq \phi^j_{s,r,c} \end{cases}, \forall i = j \quad (1)$$

As shown in (1), the behavior of a volunteer $b^i_{v,r,c}$ at i in r is assigned to zero when there is a mismatch in the observed state of channel c, between v and s.This can be because a) v makes a false detection, b) v lies about the true result, or c) s makes a false detection, d) s lies about the true result. For this paper, we assume that s is trustworthy and never makes a false detection or lies about a true result. An AUI when both v and s monitor channel c is called a matching interval. We aggregate $b^i_{v,r,c}$ over all the matching intervals to find the trust $T_{v,r,c}$ of v to monitor channel c in r, by calculating the arithmetic mean $T_{v,r,c}$, given by (2),

$$T_{v,r,c} = \frac{1}{m}\sum_{p=1}^{m} b^p_{v,r,c} \quad (2)$$

where p is a matching interval and m is the total number of matching intervals over all the monitoring intervals observed so far. After every MI, a volunteer v monitoring a channel c in region r sends the detection results (over all the AUIs in the MI) to the corresponding VS_r in the DSA Enforcement Infrastructure. Similarly, a sentinel s that monitors the spectrum in region r, sends its detection results and the random AUIs in which it monitored to VS_r. Based on the detection results of both the sentinel and the volunteers, the trust $T_{v,r,c}$ of volunteer v is computed in the VS_r. We assume that volunteers can detect spectrum misuse by using any of the methods of misuse detection used in literature [29]-[31]. The impact of choosing any of these methods for misuse detection to the accuracy of detection is out of the scope of this paper.

B. Location Likelihood

In order to efficiently support detection of channel violation in a region r, volunteers who are most likely to reside a major proportion of time in r after a visit to r, are given preference. The VS_r estimates the fraction of time that a volunteer v stays in r after its current visit to r. As shown in Figure 3, after the $(j)^{th}$ visit of v to r, we measure its $(j-1)^{th}$ sojourn time, $S_v^{j-1}(r)$, in r as the difference between its $(j-1)^{th}$ departure time, $dep_v^{j-1}(r)$ from r and its $(j-1)^{th}$ arrival time, $arr_v^{j-1}(r)$ in r. Furthermore, we calculate the $(j-1)^{th}$ return time $R_v^{j-1}(r)$ of v in r as the difference between $arr_v^{j}(r)$ and $arr_v^{j-1}(r)$. As given by (4), this enables us to calculate the proportion of time, $P_v^{j-1}(r)$, that v resided in r on its previous $((j-1)^{th})$ visit to r, as the ratio of $S_v^{j-1}(r)$ to $R_v^{j-1}(r)$. Based on this information, the VS_r estimates the proportion of time that v is likely to stay in r before its j^{th} departure from r, as an exponentially smoothed average, given by (4).

$$P_v^{j-1}(r) = \frac{S_v^{j-1}(r)}{R_v^{j-1}(r)} \quad (3)$$

$$\tilde{P}_v^{j}(r) = \alpha . P_v^{j-1}(r) + (1 - \alpha) . \tilde{P}_v^{j-1}(r) \quad (4)$$

In order to estimate the smoothed average, $\tilde{P}_{j,v,r}$ more accurately, smoothing factor α is computed as:

$$\alpha = h . \frac{(E_v^{j-1}(r))^2}{\sigma_v^{j}(r)} \quad (5)$$

where $0 < h < 1$, $E_v^{j-1}(r) = P_v^{j-1}(r) - \tilde{P}_v^{j-1}(r)$ is the prediction error, and $\sigma_v^j(r)$ is the average of the past square prediction errors on visit j. $\sigma_v^j(r)$ can be expressed as follows:

$$\sigma_v^{j}(r) = h . (E_v^{j-1}(r))^2 + (1 - h) . \sigma_v^{j-1}(r) \quad (6)$$

Moreover, at any given time t, the location $L_{v,t}$ of volunteer v enables us to estimate the likelihood of v to stay in r over the next monitoring interval, MI, based on the assumption that the likelihood of v to stay in r decreases as the displacement between $L_{v,t}$ and the centroid O_r of r increases. This is expressed by the separation factor, $\Upsilon_{t,v,r}$, given by (7) as follows:

$$\Upsilon_{t,v,r} = \gamma_1 e^{-\gamma_2 d(L_{v,t}, O_r)} \quad (7)$$

where $0 < \gamma_1, \gamma_2 < 1$, are parameters defined by the system and $d(L_{v,t}, O_r)$ is the displacement between $L_{v,t}$and O_r. Since $\Upsilon_{t,v,r}$ is exponential, so we empirically select values of γ_1 and γ_2 to avoid high variance in the values of $\Upsilon_{t,v,r}$ across all the volunteers.

Hence, the location likelihood, $L_{v,r}(MI)$ of v in r at time t over the next MI, is given by a function f of the parameters, $\tilde{P}_{j,v,r}$ of the latest (j^{th}) visit of v in r and $\Upsilon_{t,v,r}$,. We observe that since $R_{j-1,v,r} > S_{j-1,v,r}$ and $0 < \alpha < 1$, so $0 < \tilde{P}_{j,v,r} < 1$. Similarly, since $d(L_{v,t}, O_r) \geq 0$, so $0 < \Upsilon_{t,v,r} \leq 1$. As

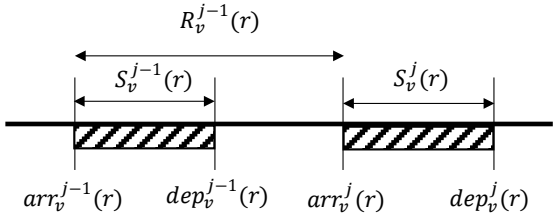


Figure 3. Sojourn time $S_{j,v,r}$ and Return time $R_{j,v,r}$ of volunteer v after its j^{th} visit to region r.

weighting the parameters by linear regression requires large amount of data and preferential weighting is hard to establish because it usually requires an expert opinion on the importance of an individual parameter relative to the overall composite parameter [17], so we assign equal weights to the parameters $\tilde{P}_{j,v,r}$ and $\Upsilon_{t,v,r}$. Finally, we define function f as the product of parameters $\tilde{P}_{j,v,r}$ and $\Upsilon_{t,v,r}$ as given by (8) below.

$$L_{v,r}(MI) = \tilde{P}_{j,v,r} \times \Upsilon_{t,v,r} \quad (8)$$

C. Selection of volunteers

From the set of volunteers, V, in area of enforcement, R, the Volunteer Selection Unit selects k_r *qualified* volunteers to monitor region r at the beginning of every MI. This is determined by the estimated Qualification $Q_{v,r,c}(MI)$ of a volunteer v to monitor a channel c in r over the next MI, given by (9), defined below.

$$Q_{v,r,c}(MI) = g(T_{v,r,c}, L_{v,r}(MI)) \quad (9)$$

Since $T_{v,r,c}$ and $L_{v,r}(MI)$ represent the measurement of different parameters, we normalize them by using the min-max normalization technique [17] such that $0 \leq T_{v,r,c}, L_{v,r}(MI) \leq 1$. Clearly, both trust and location likelihood are crucial for successful detection of spectrum access violation by crowdsourced volunteers. Therefore, we explore ways to aggregate the two parameters in g in order to assess their impact in measuring the qualification of a volunteer as shown in (10) – (13).

$$g_1 = \frac{w_1}{w_1 + w_2} p_1 + \frac{w_1}{w_1 + w_2} p_2 \quad (10)$$

$$g_2 = e^{\beta_1 . p_2} . \beta_2 . p_1 \quad (11)$$

$$g_3 = e^{\beta_1 . p_2} . \log(1 + \beta_2 . p_1) \quad (12)$$

$$g_4 = \max(p_1, p_2) \quad (13)$$

In the above equations, we assume that $p_1 = T_{v,r,c}$ and $p_2 = L_{v,r}(MI)$. In (10), we aggregate p_1 and p_2 by using weighted addition. The variant g_1 is further divided into g_{1a}, g_{1b} and g_{1c} such that $w_1 < w_2, w_1 = w_2$ and $w_1 > w_2$ respectively. In (11), we make one parameter more dominant (by having it exponentially impact the value of qualification $Q_{v,r,c}(MI)$) than the other parameter which impacts the qualification value linearly. The variant g_2 in (11) is further divided into g_{2a} and g_{2b} where we make parameter p_2 and p_1 exponentially dominating respectively. Similarly, in (12), we make one parameter more dominant by having it exponentially affect the qualification value and by having the other parameter sub-linearly (logarithmically) impact the qualification value. Likewise, we divide g_3 into g_{3a} and g_{3b}

such that p_2 and p_1 are made exponentially dominant respectively. Finally, in (13), we try the variant g_4 where the qualification $Q_{v,r,c}(MI)$ is set as the maximum of the two parameters p_1 and p_2.

This work is an extension of our previous work [1] and focuses on spectrum enforcement over multiple channels in a region. We also assume that a volunteer v can be hired to monitor more than one region over the next MI as v is mobile and can potentially cover multiple regions over a given MI. The Volunteer Selection Unit of the DSA Enforcement Infrastructure builds a centralized $||V||$-by-$||R||$ matrix $\Psi_{V,R}$, using the values of volunteer attributes from the VS_r associated with every region $r \in R$. The matrix $\Psi_{V,R}$ is a volunteer-region qualification matrix that contains the qualification values $Q_{v,r,c}(MI)$ of all $v \in V$ for every channel $c \in C$ in each region $r \in R$. The Volunteer Selection Unit selects k_r volunteers dynamically from V based on the qualification values of all $v \in V$ for every c in r, using Algorithm 1 as shown in Figure 4.

For the volunteer selection Algorithm 1, we use the volunteer-region qualification matrix $\Psi_{V,R}$ to select qualified volunteers for every $r \in R$ (line 1). At the end of a MI (line 3), the Volunteer Selection Unit gains access to the qualification values of all $v \in V$ for r from $\Psi_{V,R}$ and stores

Algorithm 1 Selection of Volunteers

1: Maintain matrix $\Psi_{V,R}$ that stores qualification values $\forall v \in V, \forall r \in R$, list of selected volunteers $V_{S,r}, \forall r \in R$
2: **for** all $r \in R$ **do**
3: **if** $t = MI$ **then**
4: $Q_r \leftarrow \Psi_{V,R}[r]$
5: **if** $k_r = 1$ **then**
6: *Run Classic Secretary Algorithm*
7: **else**
8: $m_r \leftarrow Binom(||Q_r||, 1/2)$
9: **if** $m_r > \lfloor k_r/2 \rfloor$ **then**
10: $l_r \leftarrow \lfloor k_r/2 \rfloor$
11: **else**
12: $l_r \leftarrow m_r$
13: **end if**
14: *Recursively select upto l_r volunteers*
15: $B_r \leftarrow descending\_sort(Q_r[1], ..., Q_r[m_r])$
16: $threshold \leftarrow B_r[l_r]$
17: **for** $i \leftarrow m_r + 1, ..., ||Q_r||$ **do**
18: **if** $Q_r[i] > threshold$ and $||V_{S,r}|| < k_r$ **then**
19: $V_{S,r} \leftarrow V_{S,r} \cup v$
20: **else**
21: *Reject v*
22: **end if**
23: **end for**
24: **end if**
25: **end if**
26: **end for**

Figure 4. Algorithm for selection of volunteers [19].

Algorithm 2 Assignment of Channels to Volunteers

1: Maintain Hash Table $H_{c,V_{S,r}}$ that maps a channel $c \in C$ to a list of selected volunteers $\Lambda_{c,V_{S,r}}$ ordered in descending order by their qualification values to monitor channel c, a list $V_{a,r}$ of volunteers being assigned a channel, a list of selected volunteers $V_{S,r}$
2: **for** all $r \in R$ **do**
3: **if** $t = MI$ **then**
4: **while** $||V_{a,r}|| < ||V_{S,r}||$ **do**
5: *Assign c by Round Robin to the first v in $\Lambda_{c,V_{S,r}}$*
6: $V_{a,r} \leftarrow V_{a,r} \cup v$
7: *Delete v from $\Lambda_{c,V_{S,r}}$ of all c in $H_{c,V_{S,r}}$*
8: **end while**
9: **end if**
10: **end for**

Figure 5. Algorithm for assignment of channels.

them in a list Q_r (line 4). If the number of volunteers to be selected in r, k_r is 1, then we use the classic secretary algorithm [18] to select the most qualified volunteer dynamically, with constant probability. In a classic secretary algorithm, we observe the first $||Q_r||/e$ qualification values to determine a *threshold* and then select the first of the remaining volunteers, whose qualification value is above the threshold [19]. However, if $k_r > 1$, we select volunteers dynamically by using a variant of the multiple-choice secretary algorithm, which proceeds as follows. We draw a random sample m_r from a binomial distribution $Binomial(||Q_r||, \frac{1}{2})$, from which we select up to $\lfloor k_r/2 \rfloor$ volunteers recursively (lines 8-13). We keep appending the selected volunteers in set $V_{S,r}$. If m_r is greater than $\lfloor k_r/2 \rfloor$, then we set l_r to $\lfloor k_r/2 \rfloor$, otherwise we set l_r to m_r. Next, we set a $threshold$, which is the l_r^{th} largest qualification value that we observe in the sample of first m_r qualification values. After this, we select every volunteer with qualification value greater than $threshold$, till we select a maximum of k_r volunteers (lines 16-20) [19]. We apply this algorithm for selection of volunteers in every $r \in R$.

However, this algorithm does not ensure that all the channels are covered efficiently. Thus, we develop an algorithm to efficiently assign channels to the selected volunteers as shown in Figure 5. A hash table $H_{c,V_{S,r}}(MI)$ is maintained where a channel c is mapped to the list $\Lambda_{c,V_{S,r}}$ of all $v \in V_{S,r}$ (where $V_{S,r}$ is the set of selected volunteers in region r in a MI), ordered in descending order by their qualification values to monitor channel c (line 1). For every region $r \in R$, a channel c is then assigned in a round robin manner to the topmost v in $\Lambda_{c,V_{S,r}}$, i.e., c is assigned to the volunteer most *qualified* to monitor c, after which v is deleted from the list $\Lambda_{c,V_{S,r}}$ of every channel $c \in C$ in $H_{c,V_{S,r}}(MI)$ (lines 5-7). This is continued until all the volunteers are assigned a channel to monitor. This ensures that no volunteer

monitors more than one channel over a given MI and further helps to ensure effective coverage of all channels.

V. EXPERIMENTS AND RESULTS

In this section, we discuss about the experiments that we conducted and analyze the performance of the proposed spectrum enforcement framework.

A. Simulation Environment

We simulate the enforcement framework by using the C++ version of the CSIM19 simulation engine. For simplicity, we divide the entire area of enforcement R (of total area 500,000 sq. units) into two regions of equal area. This work can, however, be easily extended to deal with more regions. With the assumption that 1 sq. unit is equivalent to 1 sq. meter and by taking the average population density of Pittsburgh (2,140/sq. km) [21], we calculate the total population (1,070 people) in the area of enforcement. A random fraction of people from the total population are chosen as volunteers (equals 183 volunteers). Volunteers are initially placed at random positions within R and they move by following the Random Waypoint Mobility Model [24] with speed ranging from 1m/s to 70m/s. The maximum speed of a volunteer is chosen higher than the usual speed limit of a vehicle in a highway in order to compensate for the limited simulation time. We assume that each region has a set of five channels to monitor. Volunteers are classified as *corrupt* and *honest*. The *corrupt* volunteers detect accurately with probability ranging from 0 to $0 + \delta$ ($\delta = 0.5$) and the *honest* volunteers detect accurately with a probability of 1. Additionally, we assume that every volunteer uses a sensing device with maximum battery capacity of 7 Wh and that the battery discharges at the rate of 1 J/s for a random time interval drawn from an exponential distribution of the mean active time interval of 100 s. After every active time interval, we assume that the sensing device remains idle for a random time interval drawn from an exponential distribution of the mean idle time interval of 10 s. The simulation runs till the battery of the sensing device used by every volunteer is exhausted, i.e., for 5610 AUIs. Each AUI is equivalent to 5 units of time and one MI is equivalent to 5 AUIs. We select $\gamma_1 = 1$ and $\gamma_2 = 0.01$ for the separation factor $\Upsilon_{t,v,r}$ of v with respect to r. Since $\Upsilon_{t,v,r}$ is exponential, so we empirically decide the value of the γ_2, which is the coefficient of $\mathrm{d}(L_{v,t}, O_r)$ from (7), to avoid high variances in the qualification values of volunteers. Furthermore, we empirically determine the values of $h = 0.03$, $\beta_1 = 10$ and $\beta_2 = 10$ in (5), (11) and (12) respectively. Finally, we assume that $k_r = k$ for every $r \in R$. The essential simulation parameters with their respective values are listed in Table I.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Area of Enforcement	$500m \times 1000m$
Population	1070
Number of Volunteers	183
Number of channels per region	5
Number of regions	2
Maximum battery capacity of a volunteer	$7\ Wh$
Number of AUIs	5610
System parameter γ_1	1
System parameter γ_2	0.01
System parameter h	0.03
System parameter β_1	10
System Parameter β_2	10

B. Metrics

We consider two primary metrics for evaluating the performance of our proposed method — the *mean accuracy of detection* and the *mean hit ratio*.

In a monitoring interval MI, if a volunteer v selected for monitoring region r has its current location in r at the beginning of an AUI, then it is a *hit*, otherwise it is a *miss* in the AUI of a MI. This is in accordance with the assumption that a selected volunteer v can successfully monitor a channel c in r over an AUI only if v resides in r over the AUI. The *hit ratio* of a region $r \in R$ over a given MI measures the ratio of the number of *hits* of all the selected volunteers to the sum of the number of *hits* and the number of *misses* of all the selected volunteers in r. A volunteer with high location likelihood will give high hit ratio. The *mean hit ratio* is computed as the average of all the *hit ratios* over all the MIs in a region. The detection of an event conducted by a volunteer is considered accurate if the detection result matches that of a sentinel s in region r at an AUI. The *mean accuracy of detection* of a volunteer is computed as the average of the number of accurate detections in a MI by the selected volunteers over the entire duration of enforcement over all the channels in a region. A volunteer with high location likelihood will give high *mean hit ratio* and a volunteer with high trust value will give high *mean accuracy of detection*.

C. Results

In Figure 6, we compare the mean hit ratio and mean accuracy of volunteers selected by using different variations of the function g that computes qualification $Q_{v,r,c}(MI)$ in (9), such that $k = 1\% - 25\%\ of\ ||V||$ and probability of a volunteer to be *corrupt* is 0.5. In the variant g_{1a}, we observe that the mean hit ratio is higher than the mean accuracy. This is because $w_1 < w_2$ (i.e., the weight associated with location likelihood $L_{v,r}(MI)$ is greater than the weight associated with trust $T_{v,r,c}$). Similarly, in the variant g_{1c}, we observe that the behavior is opposite because $w_1 > w_2$. Interestingly, in variant g_{1b}, we observe that the difference in mean accuracy and mean hit ratio (of values 0.776 and 0.822 respectively) is

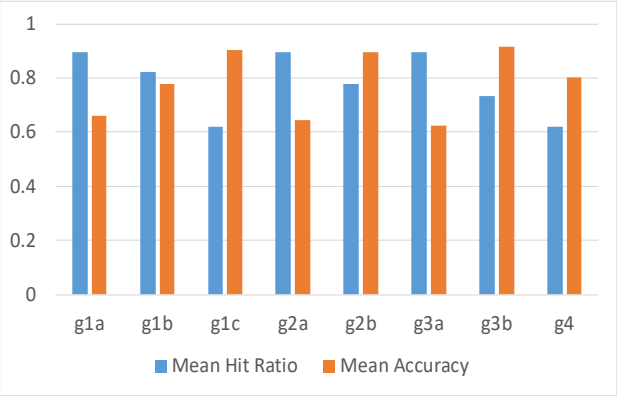


Figure 6. Comparison of the performance of volunteers selected by using different variations of function g in (9)

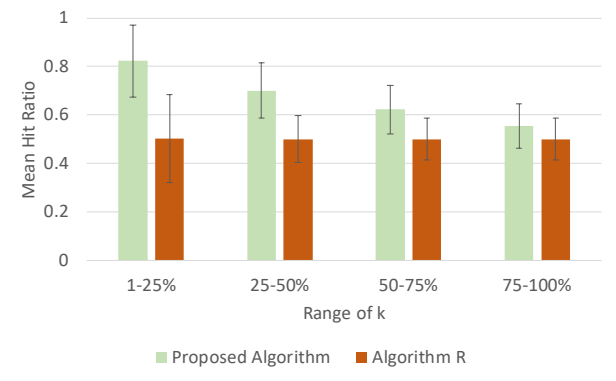


Figure 7. Comparison of the mean hit ratio of volunteers selected by using the Proposed Algorithm and Algorithm R for different values of k.

lower than what we observe in g_{1a} and g_{1c}. This is because $w_1 = w_2$ in g_{1b}. Thus, we conclude that assigning a higher weight to location likelihood results in higher mean hit ratio and assigning higher weight to trust results in higher accuracy. In the variants g_{2a} and g_{2b}, we observe that mean hit ratio is higher than mean accuracy and that mean accuracy is higher than mean hit ratio, respectively. This is because the location likelihood and trust exponentially impact the qualification value $Q_{v,r,c}(MI)$ in g_{2a} and g_{2b} respectively. We observe the same behavior in the variants g_{3a} and g_{3b}. However, we observe that the difference between mean hit ratio and mean accuracy is higher in g_{3a} and g_{3b} (of values 0.268 and 0.181 respectively) than in g_{2a} and g_{2b} (of values 0.25 and 0.119 respectively). This is because the non-dominant factor in g_{2a} and g_{2b} is linear while it is sub-linear (logarithmic) in g_{3a} and g_{3b}. Finally, for variant g_4, we observe that the mean accuracy is higher than the mean hit ratio. This is because we assume that *honest* volunteers detect accurately and hence in such cases mean accuracy is most likely to have a higher value than mean hit ratio. We want to attain both high accuracy of detection and high hit ratio. Therefore, for all the remaining experiments, we use the variant g_{1b} to calculate the qualification $Q_{v,r,c}(MI)$ as it has lowest difference (of value 0.046) between mean accuracy and mean hit ratio among all the variants of function g.

Figure 7 compares the mean *hit ratio* of all the regions over the entire duration of simulation, by using the proposed algorithm and Algorithm R for different ranges of k. Algorithm R selects k volunteers randomly from the total set of volunteers V for a region r, irrespective of their qualification. We observe that the proposed algorithm has a better mean *hit ratio* than Algorithm R for all the ranges of k. However, the mean *hit ratio* by applying the proposed algorithm decreases consistently (from 0.822 for k =1-25% of $||V||$ to 0.554 for k = 75-100% of $||V||$) with the increase in k because the proportion of highly *qualified* selected volunteers reduces as the value of k increases. The error bars in Figure 7 represent the mean standard deviation of the mean *hit ratio* across all regions, which decreases from 0.148 for k =1-25% of $||V||$ to 0.091 for k = 75-100% of $||V||$, using the proposed algorithm and decreases from 0.183 for k =1-25% of $||V||$ to 0.085 for k = 75-100% of $||V||$, using Algorithm R. This type of behavior is attributed to the fact that a balance is approached between the proportions of *qualified* and *unqualified* selected volunteers as the value of k increases.

Figure 8 compares the mean accuracy of detection of the selected volunteers over all the MIs between the proposed algorithm and the Algorithm R for varying ranges of k. We observe that the proposed algorithm performs better than the Algorithm R for all the ranges of k. The mean accuracy of detection decreases consistently (from 0.776 for k = 1-25% of $||V||$ to 0.639 for k = 75-100% of $||V||$) with the increase in k because of the decrease in the fraction of *qualified* volunteers in r as k increases. The mean standard deviation in accuracy of detection across all regions decreases from 0.124 for k =1-25% of $||V||$ to 0.049 for k = 75-100% of $||V||$, using the proposed algorithm and decreases from 0.147

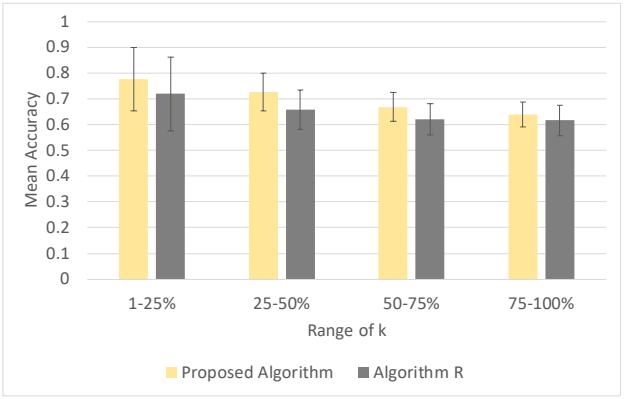


Figure 8. Comparison of the mean accuracy of volunteers selected by using the Proposed Algorithm and Algorithm R for different values of k

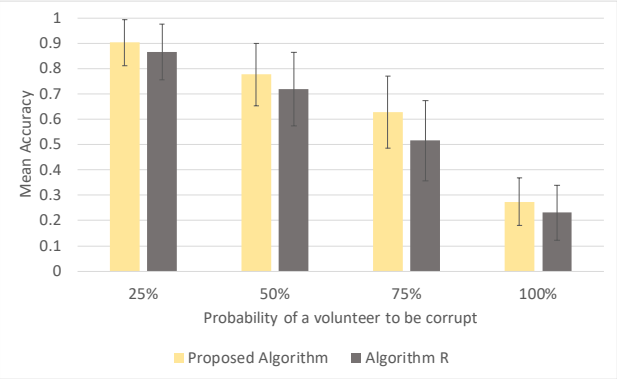


Figure 9. Comparison of the mean accuracy of detection for different probabilities of corruption of volunteers.

for k =1-25% of $||V||$ to 0.062 for k = 75-100% of $||V||$, using the Algorithm R. This is because as more volunteers are selected, a balance is approached between proportions of *corrupt* and *honest* volunteers. An interesting observation here is that the mean accuracy of selecting volunteers by using the proposed algorithm is not significantly higher than the mean accuracy attained by using Algorithm R. However, we can expect better mean accuracy by using the proposed algorithm if we use variations of g that give higher accuracy (like g_{1c}, g_{2b}, g_{3b} and g_4).

In Figure 9, we compare the mean accuracy of detection by using our proposed algorithm and Algorithm R for different probabilities of a volunteer to be corrupt. We observe that the accuracy in misuse detection decreases as the probability of a volunteer to be *corrupt* increases for $k = 1$ to 25% of $||V||$. Using our proposed algorithm, the mean accuracy decreases from 0.902 to 0.275 and by using Algorithm R, the mean accuracy decreases from 0.866 to 0.231 as the probability of a volunteer to be corrupt increases from 0.25 to 1. This is intuitive because more corrupt volunteers are selected with the increase in probability of corruption of a volunteer. Interestingly, for both the algorithms, the accuracy decreases at a faster rate than in Figure 8, proving that the probability of corruption of a volunteer has a greater impact in the overall accuracy of detection than k. Also, we observe that by using the proposed algorithm, the standard deviation increases with the increase in probability of corruption because of the increasing disparity of results between *corrupt* and *honest* volunteers. However, it decreases when the probability of a volunteer to be corrupt is 1 because of the decrease in disparity between their results (as all the volunteers are corrupt in this case).

In Figure 10, we study the mean detection accuracy across the five channels in all the regions. We observe that for $k =$ 1 to 25% of $||V||$ and the probability of corruption of a volunteer set to 0.5, the mean accuracy of detection of volunteers selected by our proposed algorithm across all channel is similar, with the highest mean accuracy of 0.833 in channel 1 and the lowest mean accuracy of 0.723 in channel 5. The standard deviation in mean accuracy across all the channels is 0.037, which is impressive. This is attributed to the efficiency of Algorithm 2 (as shown in Figure 5) that is used for the assignment of channels. However, we notice that the mean accuracy of detection decreases from channel 1 to channel 5. This is because by using Algorithm 2, the channels are assigned to volunteers in a round robin manner and hence it is more likely that a channel c_i will be assigned a more qualified volunteer than channel c_{i+1}. This discrepancy can be effectively mitigated by changing the order in which channels are assigned to volunteers after every MI. For example, if channel c_i gets assigned first to a volunteer in a MI, then channel c_{i+1} gets assigned first in the next MI. So, sequentially changing the priority of a channel to be assigned first would solve the problem.

Finally, we explore the impact of mobility pattern in the performance of volunteers for crowdsourced spectrum enforcement. We classify volunteers into three types based on their type of mobility. Volunteers of type 1 move by following the Random Waypoint Mobility model [24]. Using this model, a volunteer chooses a random destination in the area of enforcement and a random speed below the maximum speed limit to travel to the chosen destination. After reaching the destination, the volunteer pauses for a random time interval before choosing the next destination and speed. These volunteers are destination-oriented and can have speeds ranging from the speed of walking to the speed of moving in a car. The maximum speed limit is chosen to be twice the maximum speed limit of cars in USA [25], i.e., approximately 76 m/s for this type of users. The pause time of a volunteer is chosen randomly between 1 and 21 seconds. The maximum speed of a volunteer is chosen higher than the usual speed limit of a car in order to compensate for the limited simulation time. Volunteers of type 2 move in a pattern which resembles *roaming*. Type 2 volunteers choose a random direction (between 0 and 360 degrees) and move in that direction at a random speed below the maximum speed limit for a fixed interval of time. Such volunteers are assumed

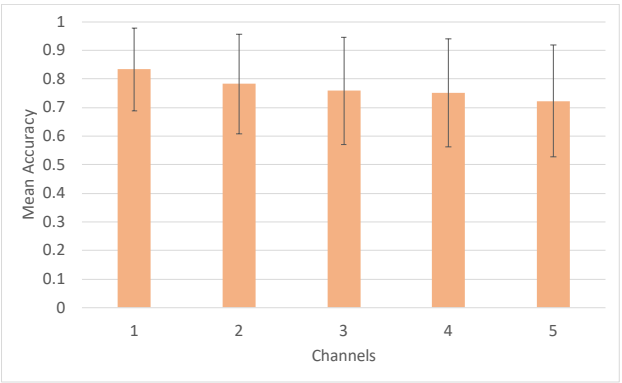


Figure 10. Comparison of the mean accuracy of detection by using the proposed algorithm across the five channels in every region.

to be walking or moving in low speed vehicles, like a skateboard and not in high speed vehicles like cars. The maximum speed limit of such volunteers is chosen as twice the average speed of a skateboarder [26], and is approximately 7 m/s. Again, the maximum speed limit of type 2 users is chosen higher than the usual speed of skateboarding in order to compensate for the limited simulation time. Type 3 volunteers are the ones whose mobility pattern is a hybrid of the mobility patterns of type 1 and type 2 volunteers. Such volunteers make a random decision to either move in a roaming pattern or by following the Random Waypoint Mobility model. After a volunteer completes its journey by using either of the mobility patterns, it will make a new random decision to again choose either of the mobility patterns for traversal.

In Figure 11, we observe the variation of mean hit ratio and mean accuracy for different mobility patterns of users for $k = 1$ to 25% of $||V||$ such that the qualification $Q_{v,r,c}(MI)$ of volunteers is calculated by using the variant g_{1b} (from (10) when w_1=w_2). We study six cases that may arise for the three types of volunteers (based on their mobility patterns). The first case arises when all the volunteers are of type 1, i.e., they follow the Random Waypoint Mobility Model (RWP). Similarly, the second and third cases are the ones where all the volunteers are of type 2 (Roaming) and type 3 (Hybrid) respectively. We observe that among the three cases when all the volunteers are either of Type 1, Type 2 or Type 3, the second case gives the highest mean hit ratio (of value 0.85) when compared to the first and third cases (of values 0.82 and 0.83 respectively). This is because the type 2 users roam at relatively lower speed ranges and hence tend to remain within the same region. Therefore, they have higher location likelihood when compared to type 1 and type 3 users. However, we observe that the mean accuracy of detection in case 2 is the lowest. This is because their location likelihood parameter dominates over their trust parameter for the calculation of their qualification values due to their high

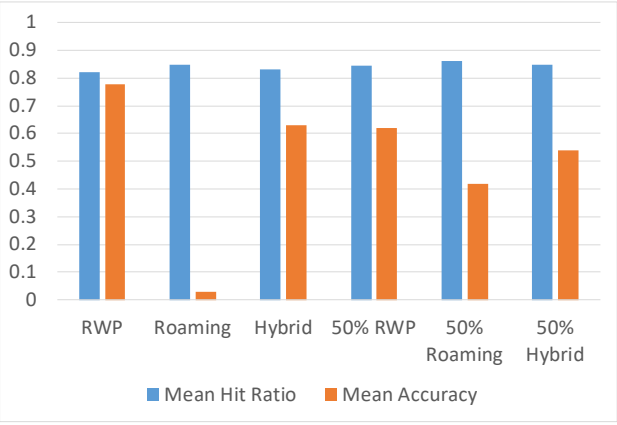


Figure 11. Variation of the mean accuracy of detection and mean hit ratio for different mobility patterns of users.

tendency to stay within the same region. Hence, even though they have high location likelihood, they are not guaranteed to give high accuracy of misuse detection. In comparison, the first and the third cases provide better accuracy of detection (of values 0.78 and 0.63 respectively). The first case provides the least difference (of value 0.043) between mean hit ratio and mean accuracy, which is desirable. Among the next three cases, the fourth case is where 50% of the volunteers follow Random Waypoint Mobility model (RWP) and the remaining volunteers are equally classified (25% each) as type 2 (Roaming) and type 3 (Hybrid) respectively. Similarly, the fifth and sixth cases are where 50% of the volunteers are of type 2 (Roaming) and type 3 (Hybrid) respectively. As expected, among these three cases, the fifth case (50% Roaming volunteers) show the highest mean hit ratio but the lowest mean accuracy. Also, we see that the fourth case (50% RWP) provide higher accuracy when compared to the sixth case (50% Hybrid). This is because hybrid volunteers do move in roaming pattern in some instances (which has previously shown lower mean accuracy). Hence, we can conclude that volunteers who move using the Random Waypoint Mobility Model with speeds ranging from the speed of walking to maximum speed limit of cars, give better performance than the other two mobility patterns because it causes least deviation between mean hit ratio and mean accuracy.

VI. CONCLUSION

In this paper, we discussed about a spectrum enforcement framework over multiple channels based on a crowdsourced spectrum monitoring infrastructure, supported by sentinel-based monitoring and a central DSA Enforcement Infrastructure. The objective was to maximize coverage of the area of enforcement, maximize coverage of channels and to ensure reliable detection of spectrum access violation by selecting highly *qualified* volunteers. We proposed to maximize the coverage of the region of enforcement by following a divide-and-conquer mechanism wherein we divide the area of enforcement into smaller regions, by applying the Lloyd's algorithm, which is a relaxation to the Voronoi algorithm. Every small region in the enforcement area is responsible for its own spectrum enforcement, which in turn ensures enforcement of the entire area. The qualification of a volunteer for the upcoming time interval is decided by its likelihood to stay in the region over the next monitoring interval and by its trust. We explored different ways to aggregate the two parameters of location likelihood and trust to find the best combination for calculating the qualification of a volunteer. We used a variant of the multiple-choice Secretary algorithm to select volunteers dynamically based on their qualifications to monitor a region. We also developed a mechanism to efficiently assign channels to the selected volunteers for monitoring. We observed the efficacy of the proposed algorithm for assignment of channels and proposed a methodology by which it can be further improved. Finally, we studied the variations in mean accuracy of detection and mean hit ratio as the probability of a volunteer

to be corrupt changes and the mobility pattern of a volunteer changes.

We plan to extend this work to explore different mechanisms to select volunteers for multi-channel spectrum enforcement. We further plan to explore machine learning based methodologies to determine the trust and location likelihood of volunteers in the enforcement area.

ACKNOWLEDGMENT

This work was sponsored in part by the National Science Foundation through grants 1265886, 1547241, 1563832, and 1642928.

REFERENCES

[1] D. Das, T. Znati, M. Weiss, P. Bustamante, M. Gomez and S. Rose, "Crowdsourced Misuse Detection in Dynamic Spectrum Sharing Wireless Networks," International Conference on Networks (ICN), 2019, pp. 74-81.

[2] Federal Communications Commission. *3.5 GHz Band / Citizens Broadband Radio Service.* [Online]. Available from: https://www.fcc.gov/wireless/bureau-divisions/broadband-division/35-ghz-band/35-ghz-band-citizens-broadband-radio#block-menu-block-4. [Accessed: 30 Aug. 2019].

[3] E. Schlager and E. Ostrom, "Property-Rights Regimes and Natural Resources: A Conceptual Analysis," Land Econ., vol. 68, no. 3, 1992, pp. 249–262.

[4] S. Shavell, "The Optimal Structure of Law Enforcement," The Journal of Law & Economics, vol. 36, no. 1, 1993, pp. 255–287. JSTOR, www.jstor.org/stable/725476.

[5] A. Gopinathan, Z. Li, and C. Wu, "Strategyproof auctions for balancing social welfare and fairness in secondary spectrum markets," 2011 Proc. IEEE INFOCOM, 2011, pp. 3020–3028.

[6] J. Lin, M. Li, D. Yang, G. Xue, and J. Tang, "Sybil-proof incentive mechanisms for crowdsensing," in IEEE INFOCOM 2017, pp. 1–9.

[7] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing," in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, 2012, pp. 173–184.

[8] M. B. H. Weiss, M. Altamimi, and M. McHenry, "Enforcement and spectrum sharing: A case study of the 1695-1710 MHz band," in 8th International Conference on Cognitive Radio Oriented Wireless Networks, 2013, pp. 7–12.

[9] D. Yang, X. Zhang, and G. Xue, "PROMISE: A framework for truthful and profit maximizing spectrum double auctions," in Proceedings - IEEE INFOCOM, 2014, pp. 109–117.

[10] R. Chen, J.-M. Park, and J. H. Reed, "Defense Against Primary User Emulation Attacks in Cognitive Radio Networks," IEEE J.Sel. A. Commun., vol. 26, no. 1, Jan. 2008, pp. 25–37.

[11] X. Jin, R. Zhang, Y. Chen, T. Li, and Y. Zhang, "DPSense: Differentially Private Crowdsourced Spectrum Sensing," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 296–307.

[12] X. Jin and Y. Zhang, "Privacy-Preserving Crowdsourced Spectrum Sensing," IEEE/ACM Trans. Netw., vol. 26, no. 3, Jun. 2018, pp. 1236–1249.

[13] A. Dutta and M. Chiang, ""See Something, Say Something" Crowdsourced Enforcement of Spectrum Policies," IEEE Trans. Wirel. Commun., vol. 15, no. 1, Jan. 2016, pp. 67–80.

[14] X. Zhu, J. An, M. Yang, L. Xiang, Q. Yang, and X. Gui, "A Fair Incentive Mechanism for Crowdsourcing in Crowd Sensing," IEEE Internet Things J., vol. 3, no. 6, Dec. 2016, pp. 1364–1372.

[15] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," ACM Comput. Surv., vol. 23, no. 3, Sep. 1991, pp. 345–405.

[16] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations," SIAM J. Numer. Anal., vol. 44, no. 1, Jan. 2006, pp. 102–119.

[17] B. Talukder, K. W. Hipel, and G. W. vanLoon, "Developing Composite Indicators for Agricultural Sustainability Assessment: Effect of Normalization and Aggregation Techniques," Resources, vol. 6, no. 4, 2017.

[18] G. Kamath. *Advanced Algorithms, Matroid Secretary Problems.* [Online]. Available from: http://www.gautamkamath.com/writings/matroidsec.pdf. [Accessed 30 Aug. 2019].

[19] R. Kleinberg, "A Multiple-choice Secretary Algorithm with Applications to Online Auctions," in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 630–631.

[20] X. Jin, J. Sun, R. Zhang, Y. Zhang, and C. Zhang, "SpecGuard: Spectrum misuse detection in dynamic spectrum access systems," 2015 IEEE Conf. Comput. Commun., 2015, pp. 172–180.

[21] Pittsburgh Population. (2018-06-12). [Online]. Available from: http://worldpopulationreview.com/us-cities/pittsburgh/. [Accessed 30 Nov. 2018].

[22] A. M. Salama, M. Li, and D. Yang, "Optimal Crowdsourced Channel Monitoring in Cognitive Radio Networks," in IEEE Global Communications Conference, GLOBECOM, Singapore, December 4-8, 2017, pp. 1–6.

[23] M. Li, D. Yang, J. Lin, M. Li, and J. Tang, "SpecWatch: A framework for adversarial spectrum monitoring with unknown statistics," Comput. Networks, vol. 143, 2018, pp. 176–190.

[24] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic Properties of the Random Waypoint Mobility Model," Wirel. Networks, vol. 10, no. 5, pp. 555–567, Sep. 2004.

[25] CNBC. *Fastest Road in America: 85 MPH and We May Be Going Even Faster.* [Online]. Available from: https://www.cnbc.com/id/49520151. [Accessed 30 Aug. 2019].

[26] RidingBoards. *Average Skateboard Speed: How Fast Do We Ride?* [Online]. Available from: https://www.ridingboards.com/average-skateboard-speed/. [Accessed 30 Aug. 2019].

[27] D. Das, T. Znati, M. Weiss, S. Rose, P. Bustamante and M. Gomez, "Spectrum Misuse Detection in Cooperative Wireless Networks," 17th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2020, in press.

[28] M. Khaledi, M. Khaledi, S. Sarkar, S. Kasera, N. Patwari, K. Derr, and S. Ramirez, "Simultaneous Power-Based Localization of Transmitters for Crowdsourced Spectrum Monitoring," 2017, pp. 235–247.

[29] S. Liu, L. J. Greenstein, W. Trappe, and Y. Chen, "Detecting anomalous spectrum usage in dynamic spectrum access networks," Ad Hoc Networks, vol. 10, no. 5, pp. 831–844, 2012.

[30] J. Tang and Y. Cheng, "Selfish misbehavior detection in 802.11 based wireless networks: An adaptive approach based on Markov decision process," 2013 Proceedings IEEE INFOCOM, Turin, 2013, pp. 1357-1365.

[31] G. Atia, A. Sahai and V. Saligrama, "Spectrum Enforcement and Liability Assignment in Cognitive Radio Systems," 2008 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, Chicago, IL, 2008, pp. 1-12.

Techniques for Enhancing the Rebalancing Algorithm for Folded Clos Networks

Satoru Ohta
Department of Electrical and Computer Engineering, Faculty of Engineering
Toyama Prefectural University
Imizu, Toyama, Japan
e-mail: ohta@pu-toyama.ac.jp

***Abstract*—Folded Clos Networks (FCNs) are switching networks constructed by connecting small switches aligned to ($2p + 1$)-stages ($p = 1, 2, 3, \ldots$). FCNs have often been examined in previous studies on data center networks. To take advantage of the high bandwidth provided by an FCN, it is necessary to establish an adequate routing method that uniformly diffuses flows. A previous study proposed the rebalancing algorithm as such a method, which is executable with locally obtainable information at each switch. By applying the rebalancing algorithm to an FCN, it becomes possible to impose an upper bound on the number of flows passing through a link. This means that the rebalancing algorithm prevents the link load from becoming excessively high. This paper reviews theoretical aspects of the algorithm applied to three- and five-stage FCNs. Then, two techniques are proposed for improving the rebalancing algorithm in terms of the load equality between links. The techniques distribute traffic more uniformly and do not affect the upper bound on the number of flows on a link. The effectiveness of the two techniques is assessed via computer simulation for different traffic and network models. The network models include a three-stage FCN and a five-stage FCN. The simulation results demonstrate the effectiveness of the two proposed techniques.**

Keywords- network; algorithm; routing; data center; packet.

I. INTRODUCTION

Data center networks are becoming increasingly important, as the majority of popular information services are provided via data centers. It is thus essential to establish topologies for high-performance data center networks. To this end, studies on data center networks have been performed based on several topologies, including the Clos network [1], [2] fat-tree [3], DCell [4], and BCube [5]. Among these topologies, the Clos network has often been studied because it functions as a scalable and high-bandwidth network by interconnecting small commodity switches. Various data center networks based on the Clos network topology have been implemented [2], [6]–[8].

A Clos network is a three-stage nonblocking switching network originally developed by Charles Clos in 1953 [9]. On the basis of this three-stage network, it is possible to configure nonblocking switching networks with ($2p + 1$)-stages ($p = 1, 2, 3, \ldots$). In data center network applications, the network appears in the form of a Folded Clos Network (FCN). A three-stage FCN is roughly equivalent to a three-stage network; however, it is constructed by folding the corresponding three-stage Clos network at its center. Similarly, it is also possible to construct a five- or seven-stage FCN.

To apply an FCN to data center networks, the routing of a packet is important. Inadequate routing may cause a load imbalance between the links, which in turn may cause traffic congestion and degrade performance. However, if the load is uniformly distributed among the links, an FCN can achieve high throughput by fully utilizing the bandwidth of every link.

Several previous studies [7], [8], [10] have used a routing method that involves forwarding a packet to a randomly selected route. This method is reasonable, as it uniformly distributes the average number of flows between links. However, using this method, there is a high probability that the load on a given link will become excessively large. Consequently, traffic congestion may occur due to heavily loaded links, thus degrading network performance. As discussed in [11], this problem may be critical for big data applications, which require high-bandwidth transmission. It is therefore important to develop a routing algorithm that diffuses the traffic load more uniformly than random routing.

A routing algorithm for an FCN should be executable in a distributed manner to decrease the processing overhead generated by handling frequent route decisions. In addition, the algorithm should function without global information of the entire network to eliminate the communication overhead associated with gathering information.

Routing can be performed on either a per-packet or per-flow basis. The former method determines a route in a packet-by-packet manner. Thus, packets that belong to the same flow may pass through different routes. Since delays are also different depending on the routes, packet reordering occurs for per-packet routing. Meanwhile, the latter method determines a unique route for a flow. Every packet of the flow goes through that route. This study examines a method based on per-flow routing because packet reordering is unavoidable for per-packet routing.

Ohta [12] presented two distributed algorithms – the rebalancing algorithm and the load sum algorithm – that diffuse flows in FCNs. Using computer simulations, it was demonstrated that these methods diffuse flows more uniformly than random routing. Of the two methods, the rebalancing algorithm uses information that is locally obtainable at the source switch of a flow. Although the load sum algorithm has superior performance with respect to load equality, it is less practical due to the communication overhead between switches. Thus, if the rebalancing

algorithm is improved to diffuse flows more uniformly, a more practical and efficient algorithm can be obtained.

In [1], Ohta proposed techniques for improving the rebalancing algorithm with respect to load equality. These techniques are based on information that is locally obtainable at the source switch of a flow. The first technique modifies the algorithm to distribute uplink loads more evenly, while the second technique utilizes the fact that the algorithm has a process for scanning middle switch indices for routing and rerouting. Therefore, using the second method, the order of scanning the middle switch indices is determined to uniformly diffuse flows.

An advantage of the rebalancing algorithm is that an upper bound is theoretically derived for the number of flows on a link. When using the abovementioned improvement techniques, this upper bound is not affected. Therefore, the worst-case link load is limited, as in the case in which these techniques are not applied. The effectiveness of the two techniques was confirmed via computer simulations.

This paper expands the work of [1] and offers the following contributions:

- A discussion of the presented techniques applied to a five-stage FCN.
- Performance evaluation under traffic conditions not considered in [1].

A theoretical upper bound is derived for the number of flows on a link for a five-stage FCN as well as a three-stage FCN. The performance of the proposed techniques applied to a five-stage FCN is also evaluated through computer simulation. With respect to traffic conditions, this paper examines several different traffic models in simulation. In one traffic model, flows are equally generated for every pair of source and destination switches. Other traffic models are lightly or heavily skewed. For these traffic conditions, flows are destined to a limited number of destination switches from a certain source switch. A traffic model that generates light load is also examined. The simulation results reveal that one of the proposed techniques is not effective for a highly skewed traffic model. However, this technique is more effective for light traffic load. The load equality is effectively improved for every examined traffic model if both proposed methods are used simultaneously.

The remainder of this paper is organized as follows. Section II provides a discussion of the FCN, while Section III reviews related work. Section IV explains the rebalancing algorithm, while Section V presents two modification techniques. Section VI evaluates the effectiveness of the techniques, and Section VII concludes the paper.

II. FOLDED CLOS NETWORK

Figure 1 presents an example of a Clos network. As illustrated, the network is a three-stage switching network that consists of r first-stage switches, m second-stage switches, and r third-stage switches. Each first-stage switch has n input ports whereas each third-stage switch has n output ports. This switching network was originally proposed by Charles Clos [9] and has been comprehensively investigated over a long period of time [13].

It is possible to construct a five-stage Clos network by replacing the second-stage switches with three-stage Clos networks. For example, consider the case where $n = m = 2$ and $r = 6$ in the configuration in Figure 1. Then, by replacing the second-stage switch with a three-stage network of $n = m = 2$ and $r = 3$, a five-stage network is obtained, as illustrated in Figure 2. By repeating this procedure, it is possible to construct a $(2p + 1)$-stage Clos network for any $p > 1$.

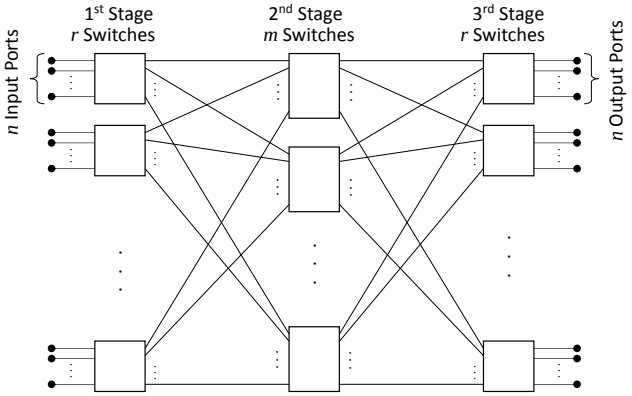


Figure 1. Example of a three-stage Clos network.

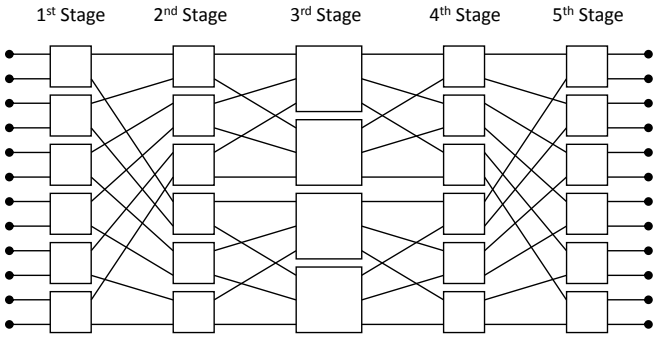


Figure 2. Example of a five-stage Clos network.

The advantages of the Clos network are its small amount of hardware and nonblocking nature. To demonstrate its advantages in hardware amount, let N denote the total number of input or output ports. In the classical switching network theory, the hardware amount of a switching network is often measured by the number of crosspoints, assuming that each small switch is a crossbar. A single crossbar switch with N ports has N^2 crosspoints, and the number of crosspoints is $O(N^{3/2})$ for a three-stage Clos network [9]. Thus, the amount of hardware is much smaller for a Clos network than for a single crossbar switch. In addition, the number of crosspoints is $O(N^{4/3})$ for a five-stage Clos network. Thus, a five-stage configuration can be constructed with less hardware for a larger value of N. It is also known that the amount of hardware for a Clos network is $O(Ne^{2\sqrt{(\log_e 2)\cdot(\log_e N)}})$ when the number of stages is optimized for N [14].

A three-stage FCN is roughly equivalent to a three-stage Clos network. However, an FCN is constructed by folding the three-stage network at its center. An example of a three-stage

FCN is provided in Figure 3. In a three-stage FCN, the first- and third-stage switches are integrated into input/output switches, and the second-stage switches are middle switches that connect the input/output switches.

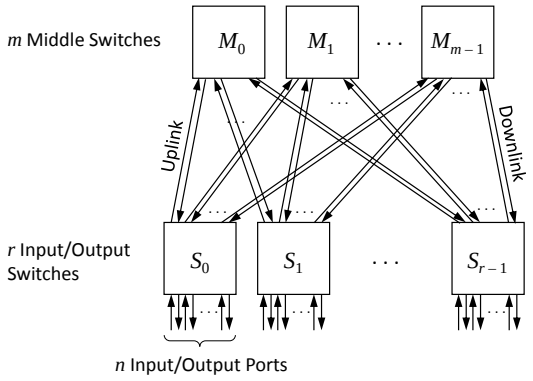


Figure 3. Example of a three-stage FCN.

As illustrated in Figure 3, r input/output switches are labeled S_0, S_1, …, S_{r-1}, while m middle switches are labeled M_0, M_1, …, M_{m-1}. Every middle switch is connected to every input/output switch via an uplink and downlink. An uplink is set from an input/output switch to a middle switch, whereas a downlink is set in the reverse direction. Each input/output switch has n input/output ports.

By implementing each switch as an IP (Internet Protocol) layer 2 or 3 switch, a data center network can be constructed on the basis of an FCN. It is known that a data center network based on Clos topology has the advantages of scalability and high bandwidth.

Because this paper considers applications to data center networks, the information passes through the FCN via packets. Since a middle switch is connected to every input/output switch, a packet can reach its destination switch from an arbitrary middle switch via a downlink. Therefore, the source switch can transmit a packet to the destination switch via any middle switch. However, the traffic load on an uplink or downlink depends on the routing at the source switch. If the routing is inadequate, traffic congestion occurs, which degrades performance. Congestion can be avoided if the traffic is evenly diffused between the uplinks and downlinks in an FCN. It is therefore important to establish a routing method that is executed at the source switch of a packet.

This paper assumes that routing is performed on a flow basis. A flow is a packet stream identified by a set of fields in the packet header [15]. A frequently used field set is {*source address*, *destination address*, *protocol*, *source port*, *destination port*}, which is associated with an IP socket. A different field set can also be used as flow identifiers. If a fixed route is assigned to a flow, packet reordering does not occur. This is advantageous because packet reordering leads to throughput degradation. This paper considers the case in which an FCN connects many hosts and processes via its $N = nr$ input/output ports. In this situation, many concurrent flows exist between ports.

Flows are categorized as elastic or stream [16] depending on the nature of the traffic. In this paper, it is assumed that the network handles elastic flows. This assumption is reasonable because many services are supported by TCP (Transmission Control Protocol), and TCP traffic is elastic. For elastic flows, the throughput of a flow is restricted by the link capacity portion shared with other flows. Therefore, to achieve high throughput, it is indispensable to uniformly diffuse the number of flows among links and decrease the maximum number of flows on a link. Thus, in this paper, the link load is assessed by the number of flows.

Whereas the network illustrated in Figure 3 is based on a three-stage Clos network, it is also possible to construct an FCN from a five-stage Clos network. Figure 4 presents an example of a five-stage FCN. This network is the configuration that results from replacing a middle switch with an FCN in the configuration in Figure 3. This type of five-stage FCN is used as a data center network, as reported in [6].

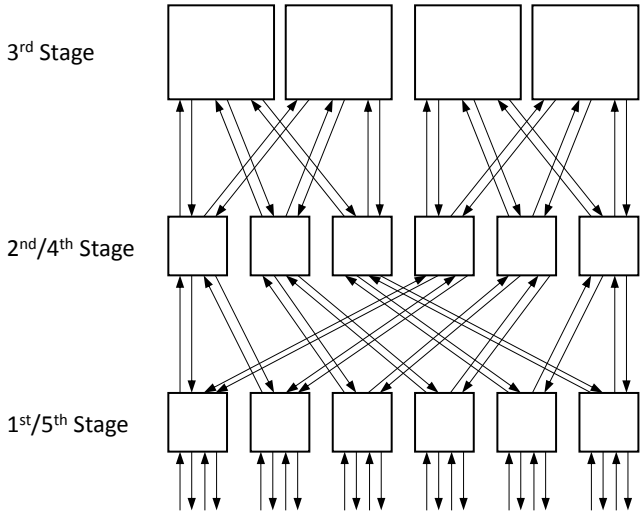


Figure 4. Example of a five-stage FCN.

III. Related Work

An FCN first appeared in the original work by Clos [9] as the triangular array configuration. The configuration has also been referred to as a Clos truncated network [17] or single-sided Clos network [18]. Clos networks or FCNs have been used for various applications including telephone switching [9], [17] cross-connect systems [19]–[21], multi-processor computers [10], network-on-chip [22]–[24], and data center networks [1], [2], [6]–[8], [11], [12], [25]–[28].

For computer network applications, including data center networks, Clos networks are treated as packet switching networks. Various studies have investigated the application of Clos networks as packet switching networks. The routing methods examined in those studies are categorized as per-packet routing and per-flow routing. As an example of per-packet routing, Hassen and Mhamdi [25] investigated a Clos network that had crossbar switches with small-size buffers in each stage. For this configuration, Hassen and Mhamdi proposed distributed and centralized packet scheduling

mechanisms to achieve low packet delay and high throughput. However, their proposed method involves per-packet processing to perform scheduling. Thus, the processing load generated by each packet arrival will become critical for a large network that handles a large number of packets in a unit time. In addition, the advantage of a small-size buffer may not be significant because buffer memory is not expensive.

In another study, Hassen and Mhamdi [26] investigated a modified Clos network configuration that employed a multi-directional network on a chip as a switching module in each stage. This module provides interconnections between middle stage switching modules. For this network configuration, inter-module routing is performed based on global network congestion information. However, it is uncertain whether gathering congestion information for routing is practical.

A quite different per-packet routing method was proposed by Yang et al. [27]. This method determines the switch connections in each stage from a given traffic matrix, where each element of the matrix represents the number of packets to be transmitted from a source to a destination during a certain period. The connections are reconfigured several times to minimize the cost during the transmission period. Unfortunately, it is very unlikely that an exact traffic matrix could be obtained in a real-world data center network. Thus, it may be difficult to apply their method.

To avoid congestion in a Clos network, Ghorbani et al. [28] proposed a method that diffuses traffic on a packet-by-packet basis. This method determines the next hop of a packet according to the queue length of each output buffer in a switch. Consequently, packets of a flow may go through routes with different queueing delays, and this delay variation may cause packet reordering. Ghorbani et al. performed a computer simulation and observed that 0.02 % of packets were delivered out of order. The extent to which this rate of packet reordering affects TCP throughput is unclear.

The studies of [25]–[28] are theoretical, and the proposed methods are evaluated through computer simulations. However, an actual implementation of per-packet routing was reported by Scott et al. [10]. Their Clos network is designed for the interconnection of a multi-processor high performance computer. The routing method employed in their work determines the next hop of a packet depending on the buffer space. The proposed method is based on a custom packet format developed for the multi-processor computer. Consequently, their method will not be directly applicable to datacenter networks.

A more practical approach to congestion avoidance is traffic diffusion based on per-flow routing methods. A common method of flow-based traffic diffusion involves routing a flow to a randomly selected middle switch. This technique, which is referred to as Valiant load balancing, is employed in the system implemented by Greenberg et al. [7] and was originally proposed by Valiant [29] for a binary cube topology. Al-Fares et al. [8] explored a method of computing routing table entries from the indices of switches and host identifiers. This is equivalent to randomly assigning route flows using the output of a hash function fed by switch indices and host identifiers. The architecture reported by Scott et al. employs a per-packet adaptive routing mechanism as well as per-flow deterministic routing [10]. For deterministic routing, flows are diffused to random middle switches via a hash function fed by input ports and destinations.

The idea of randomly routing flows is reasonable because the average number of flows is balanced between links. However, there is a high probability that the worst-case load on a certain link will become excessively large. This can cause traffic congestion and degrade performance, such as packet latency or network throughput. The adaptive routing proposed by Zahavi et al. [11] may reduce this disadvantage of random routing. Initially, this method semi-randomly selects routes for the flows at the source switches. Then, the destination switches identify bad links, which are excessively loaded by the initial routing. Next, the destination switches notify the source switches of the flows, passing the bad links as bad flows. With this notification, the source switches can reroute the bad flows, and the rerouting is repeated until there are no bad links. In [11], the convergence of the rerouting was evaluated using a theoretical analysis based on Markov chain models and computer simulation. However, the number of times the flows must be rerouted to eliminate all bad links in the worst case is not theoretically known. It is also unclear whether bad links are definitively removed by Zahavi et al.'s method. As a result, this method may not be practical.

Ohta [12] presented two flow-based routing methods that diffuse flows more uniformly than random routing. Using these methods, a flow is routed (or rerouted) at its source switch in a distributed manner. One of these methods is the rebalancing algorithm, which runs using locally obtainable information. The other method is the load sum algorithm, which requires communication between source and destination switches. Simulation results demonstrate that these methods both outperform random routing. In addition, the load sum algorithm is shown to diffuse flows more uniformly than the rebalancing algorithm.

The simulation results reported in [12] indicate that all flow equality metrics are smaller for the load sum algorithm than for the rebalancing algorithm. However, the load sum algorithm may be inefficient with respect to the communication overhead between switches, in particular in the case of short-duration flows. Namely, the amount of traffic exchanged by a short-duration flow may become comparable to or smaller than that exchanged by the communication between switches, which is highly inefficient. From this viewpoint, the rebalancing algorithm is likely to be more practical. With improvements to this method in terms of the uniformity of flow diffusion, the rebalancing algorithm can become more effective.

IV. REBALANCING ALGORITHM

This paper focuses on the rebalancing algorithm presented in [12]. This algorithm is in fact a packet stream version of the method described in [30]. It assumes that the route of a newly generated flow is determined when its first packet arrives at the input switch. Implementing such a mechanism with currently available technology would not be simple; however, it is important to investigate potential methods that display higher performance than conventional routing.

This section presents several definitions, specifies local information, and outlines the algorithm.

A. Definitions

Throughout this paper, the following variables are used for a three-stage FCN:

- $F(i, j, k)$: number of flows that pass through a source switch S_i, middle switch M_j, and destination switch S_k $(0 \le i, k \le r-1, 0 \le j \le m-1)$
- $U(i, j)$: number of flows on the uplink set from S_i to M_j
- $D(j, k)$: number of flows on the downlink set from M_j to S_k

In an FCN, if the source switch of a flow is the same as its destination, it is not necessary to route the flow to a middle switch. The flow can be directly routed to the destination within the source/destination switch. In view of this characteristic, $U(i, j)$ and $D(j, k)$ are related to $F(i, j, k)$ as follows:

$$U(i, j) = \sum_{k=0, i \neq k}^{r-1} F(i, j, k), \quad (1)$$

$$D(j, k) = \sum_{i=0, i \neq k}^{r-1} F(i, j, k). \quad (2)$$

The algorithm is described using these variables. Table I summarizes the symbols used in this paper.

TABLE I. TABLE OF SYMBOLS.

Symbol	Definition
m	Number of middle switches in a three-stage FCN
n	Number of input/output ports that an input/output switch has in a three-stage FCN
r	Number of input/output switches in a three-stage FCN
m_1	Number of sub-FCNs in a five-stage FCN
n_1	Number of input/output ports that a 1st/5th stage switch has in a five-stage FCN
r_1	Number of 1st/5th stage switches in a five-stage FCN
m_2	Number of 3rd srage switches in a sub-FCN of a five-stage FCN
n_2	Number of input/output ports that a 2nd/4th stage switch has in a five-stage FCN
r_2	The number of 2nd/4th stage switches in a sub-FCN of a five-stage FCN
S_i	Input/output switch of a three-stage FCN
M_j	Middle switch of a three-stage FCN
$F(i, j, k)$	Number of flows established from S_i to S_k via M_j
$U(i, j)$	Number of flows on the uplink from S_i to M_j
$D(j, k)$	Number of flows on the downlink from M_j to S_k
α	Positive integer used
f_0	Maximum number of flows for an input/output port
f_1	Maximum number of flows on a link between an input/output (1st/5th stage) switch and a middle (2nd/4th stage) switch for a three-stage FCN (five-stage FCN)
f_2	Maximum number of flows on a link between a 2nd/4th stage switch and a 3rd srage switch for a five-stage FCN

B. Locally obtainable information

For data center network applications, flows may be frequently generated and completed in the FCN. In this situation, the routing of a flow should be executed in a distributed manner because the load resulting from frequent route decisions becomes excessively high for concentrated computations. In addition, it is impractical to perform communication between switches because there may be very short flows consisting of only several packets. As described in Section III, it is inefficient to exchange packets between switches for the routing of such short flows. Therefore, the route of a flow should be determined at its source switch using locally obtainable information.

Let us consider the case in which the FCN is a three-stage configuration. An input/output switch can obtain the headers of the packets, which arrive from its input port and are forwarded to middle switches. From these headers, the switch can identify the flows to which the packets belong. Because the switch determines the routes for the flows at the source, it can count the number of flows that travel from itself to each middle switch. Therefore, $U(i, j)$ can be managed at the source switch S_i. In addition, the switch can extract the destination switch of the flows from the packet headers. Using this information, the source switch S_i can also count $F(i, j, k)$.

Suppose that a new flow is generated and that its source switch is S_i. Then, assume that S_i can detect the arrival of a new flow. This is possible by comparing the flow identifiers to the routing table. It is also possible for S_i to detect the completion of a flow by a timeout. Therefore, S_i can launch routing or rerouting processes at flow arrival or completion.

C. Algorithm properties

The rebalancing algorithm is detailed in [12]. The algorithm utilizes parameter α, positive integer that controls its behavior. The rebalancing algorithm has the following property:

Property 1: *With the rebalancing algorithm,*

$$F(i, j, k) \le F(i, j', k) + \alpha, \quad (3)$$

for $0 \le j, j' \le m-1,\ 0 \le i, k \le r-1.$

The proof for Property 1 is found in [12]. An advantage of the rebalancing algorithm is that an upper bound exists for the number of flows on an uplink or downlink. Let f_0 denote the maximum number of flows for an input or output port. In addition, let f_1 denote the number of flows on an uplink and downlink. Then, the following property is obtained [12]:

Property 2: *When the rebalancing algorithm is performed on a three-stage FCN characterized by parameters m, n, and r,*

$$f_1 \le \frac{nf_0}{m} + \alpha\left(1 - \frac{1}{m}\right)(r-1) \quad (4)$$

Property 2 is proved from (1), (2), and (3) via the method outlined in [12]. This property ensures that the load on a link does not become very high.

In the rebalancing algorithm, parameter α determines the frequency of rerouting as well as the uniformity of flow diffusion. If α is large, rerouting never occurs. In this case, flows are diffused via route decision when they arrive at their source switches. Simulation results demonstrate that the algorithm works well even without rerouting. Following [12], a rebalancing algorithm that omits the rerouting process is hereafter referred to as *balancing algorithm*.

D. Rebalancing algorithm for five-stage FCNs

The rebalancing or balancing algorithm can be also applied to five-stage FCNs. This subsection shows that the number of flows on a link is upper bounded by the rebalancing algorithm for a five-stage FCN as well as for a three-stage FCN. The characteristic of the rebalancing algorithm applied to a five-stage FCN has never been reported elsewhere. Thus, the analysis on the five-stage FCN case is a main contribution of this paper.

As illustrated in Figure 5, a five-stage FCN can be seen as a combination of three-stage FCNs. In Figure 5, the second/fourth-stage switches and third-stage switches configure m_1 sub-FCNs, 0, 1, …, $m_1 - 1$. By considering these sub-FCNs as m_1 switches, the algorithm can be executed at the first/fifth-stage switches. Within each sub-FCN, it is also possible to run the algorithm at each second/fourth-stage switch. For this scheme, it should be noted that flow rerouting between first/fifth-stage switches causes a new flow generation and flow completion in the affected sub-FCNs. This means that flow rerouting may generate additional flow rerouting in the sub-FCN.

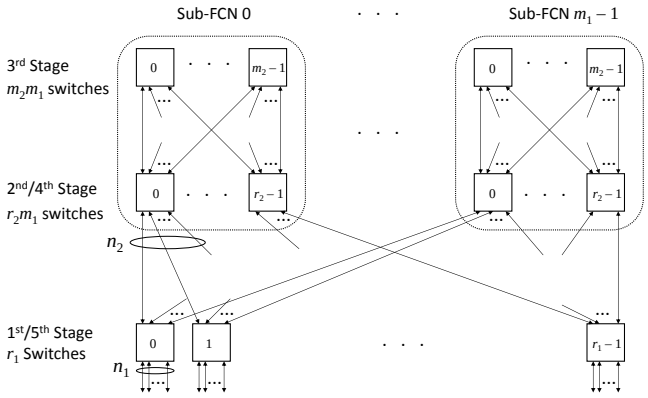


Figure 5. Parameters m_1, m_2, n_1, n_2, r_1, and r_2 for a five-stage FCN.

In the five-stage FCN case, the number of flows is upper bounded by employing the rebalancing algorithm as well as in the three-stage FCN case. Let n_1 denote the number of input/output ports of a first/fifth-stage switch, and let n_2 denote the number of input/output ports of a second/fourth-stage switch. In addition, let r_1 be the total number of first/fifth-stage switches. Assume that each sub-FCN is constructed by m_2 third-stage switches and r_2 second/fourth-stage switches. For this configuration, r_1 must be equal to r_2n_2.

Let f_1 be the maximum number of flows on an uplink or downlink between a first/fifth-stage switch and a second/fourth-stage switch. Additionally, let f_0 denote the maximum number of flows for an input or output port. Then, from Property 2,

$$f_1 \le \frac{n_1 f_0}{m_1} + \alpha\left(1 - \frac{1}{m_1}\right)(r_1 - 1) \quad (5)$$

Next, let f_2 be the maximum number of flows on a link between a second/fourth-stage switch and third-stage switch. Then, because a sub-FCN is also controlled by the rebalancing algorithm, f_2 is bounded by f_1 as follows:

$$f_2 \le \frac{n_2 f_1}{m_2} + \alpha\left(1 - \frac{1}{m_2}\right)(r_2 - 1) \quad (6)$$

Then, from (5) and (6),

$$f_2 \le \frac{n_1 n_2 f_0}{m_1 m_2} + \frac{n_2 \alpha}{m_2}\left(1 - \frac{1}{m_1}\right)(r_1 - 1) + \alpha\left(1 - \frac{1}{m_2}\right)(r_2 - 1) \quad (7)$$

Thus, the load on every link is also upper bounded for five-stage FCNs.

V. Modification Techniques

This section presents two modification techniques to improve the load equality of the rebalancing algorithm. These techniques add criteria for selecting the middle switch index. However, they do not change the conditions that $F(i, j, k)$s must satisfy. Therefore, (3) holds even if these techniques are applied. Consequently, the upper bound expressed by (4) or (7) is unchanged by these techniques.

A. Uplink flow diffusion

The rebalancing algorithm uses $F(i, j, k)$ and flow arrival and completion events from the local information. Therefore, of the available local information, $U(i, j)$ remains unused. Although the rebalancing algorithm decreases the difference between the $F(i, j, k)$s for a particular pair of i and k, the uplink load $U(i, j)$ is not necessarily uniformly distributed. For a new flow arrival, the middle switch M_J is selected such that $F(i, J, k)$ is the minimum of the $F(i, j, k)$s. In this process, there may be two or more candidates for J. Let us select J from the candidates so that $U(i, J)$ is the minimum of the candidates. Then, flows are more uniformly distributed between the uplinks. This does not necessarily improve the load equality between the downlinks; however, the performance is improved for the uplinks.

Similarly, flow diffusion via rerouting can also be modified using $U(i, j)$. For rerouting, middle switch M_J is selected such that $F(i, J, k)$ is the maximum of the $F(i, j, k)$s. Suppose that there are two or more such indices J. Then, it is possible to use the index that maximizes $U(i, J)$. We refer to this modification using $U(i, j)$ as *modification 1*.

B. Start index for scanning the middle switches

The order of searching for middle switch index J also affects the performance of the rebalancing algorithm. Assume that J is scanned in the order of 0, 1, …, $m - 1$ for a new flow

arrival. Then, a smaller index is more likely to be selected as J. Therefore, $F(i, j, k)$ has a high probability of being larger for a smaller index j even though the differences between the $F(i, j, k)$s are bounded by α for fixed i and k. According to (1) and (2), this implies that $U(i, j)$ and $D(j, k)$ also tend to be larger for a smaller value of j. To avoid this imbalance between $U(i, j)$ and $D(j, k)$, the scanning of middle switches should start from a different index depending on k for a fixed value of i. Similarly, the start index should differ depending on i for a fixed value of k. In addition, the start index should be evenly distributed between 0, 1, …, $m-1$ for different values of i or k. To satisfy this requirement, let us examine the following start index j_s:

$$j_s = (i + k)\lceil m / r \rceil . \quad (8)$$

If j_s is greater than $m-1$, j_s is replaced by j_s mod m. In (8), the term $\lceil m / r \rceil$ is necessary for evenly distributing j_s between 0, 1, …, $m-1$ for the case of $m \geq 2r$. For a new flow arrival, the index is scanned in the order of j_s, $j_s + 1$, $j_s + 2$, …; if the index reaches m, it wraps to 0.

For rerouting, simulation results reveal that the index should be started from $(j_s + m)$ mod m and then decreased. If the index reaches -1, it wraps to $m-1$. The rationale for this scheme is as follows. The scheme aims to generate a situation in which $F(i, j_s, k) \geq F(i, j_s + 1, k) \geq F(i, j_s + 2, k) \geq \ldots$. To maintain this situation, it is preferable to select J from later elements of the sequence j_s, $j_s + 1$, $j_s + 2$, …, $(j_s + m)$ mod m because $F(i, J, k)$ decreases due to rerouting. The use of the abovementioned start index is hereafter referred to as *modification 2*.

VI. EVALUATION

The effectiveness of the improvements was evaluated using computer simulations. The simulations examined the rebalancing and balancing algorithms to which modifications 1 and 2 were applied. For comparison, the original rebalancing and balancing algorithms reported in [12] were also evaluated. In the rebalancing algorithm, parameter α was set to 1.

A. Load equality metric

The degree of load equality was estimated using the following metrics, which were also used in [12]:

- Maximum: the maximum number of flows in the links at a certain measurement time
- Variance: the variance in the flow numbers in the links at a certain measurement time
- Bad links: the number of links in which the number of flows exceeds threshold C at a certain measurement time

By using three different metrics, it is possible to reliably evaluate load equality. Of the metrics, it is evident that variance is an adequate measure for load equality. However, even for the same variance or standard deviation value, the degree of traffic congestion may differ. For example, consider two cases that exhibit the same variance value. For these cases, suppose that the maximum number of flows passing through a link is greater for one case than the other. Then, because the flows share the link capacity, the throughput of a flow will decrease and the performance will be more strongly degraded in the former case. This suggests that the maximum metric is necessary as well.

It is also evident that the employment of the variance and maximum metrics are insufficient. Consider two cases with an identical maximum metric value. The first case is a situation in which the number of flows takes the maximum value for only one link but is not large for any other links. In contrast, in the second case, the number of flows is equal or close to the maximum value for many links. It is clear that a greater number of flows will be degraded in the latter case. That is, the range of flows affected by congestion differs for these cases. The bad links metric is thus essential for distinguishing this difference.

B. Simulation model

In the simulations, the following two network models were employed:

- Three-stage FCN with $r = 48$, $m = n = 24$, and
- Five-stage FCN: $r_1 = 144$, $m_1 = n_1 = 8$, and $m_2 = n_2 = r_2 = 12$.

The parameters used for the three-stage FCN were the same as those used in the model examined in [11]. Thus, the parameters were adequate for simulating a realistic network. The parameters for the five-stage FCN model were determined so that the same number of links as in the three-stage model were generated between the stages. Thus, for both models, there were $m \times r = m_1 \times r_1 = m_1 \times m_2 \times r_2 = 1{,}152$ uplinks and downlinks between stages. The total number of input or output ports was also 1,152 ($= r \times n = r_1 \times n_1$).

A flow was generated by opening a socket between hosts a and z, which were connected to two different input/output switches. By opening a socket, two flows were generated in the direction from a to z as well as in the reverse direction.

The simulation examined five traffic models. These models were constructed as follows. A previous study [7] reported that, in a real-world data center, an average machine has 10 concurrent flows. By aggregating the traffic from 10 such machines, the average number of flows is 100 for a port on each input/output switch. Four traffic models simulated this situation, and one traffic model simulated a lighter load, i.e., 25 flows on average for each input/output port.

The average number of flows was set to 100 or 25 for a port as follows. The duration of a socket was a random value according to an exponential distribution with an average of 57.6 s. This value is realistic to some extent because 10's of gigabytes of data are transmitted for some "big-data" applications [11]. The transfer time will reach some ten seconds for such applications even if the flow throughput reaches some Gb/s. To set the average number of flows to 100, the interval of opening sockets was randomly determined by an exponential distribution at an average of 0.001 s. Now, let N flows exist at a certain time in the FCN. Then, on average, N/57.6 flows are completed in 1 s whereas 2/0.001 flows are generated in 1 s. In equilibrium, these flow completion, and

generation rates are balanced. This suggest that N equals $2 \times 57.6/0.001$. Thus, the average number of flows provided to one of 1152 input or an output port is 100. Similarly, by setting the average interval of opening sockets to 0.004 s, the average number of flows to a port is 25.

The threshold for bad links, C, was set to 105 for the models, where the average number of flows given to a port is 100. When the average number of flows is 25, C was set to 30. These values were slightly larger than the average number of flows for the given traffic condition. The values of the above metrics would have been smaller if the flows were more uniformly diffused.

Four traffic models are the same in point that the average number of flows given to a port is 100. Of the models, three are intended for the three-stage FCN, while one model is intended for the five-stage FCN. The models intended for the three-stage FCN differ in their selection of source-destination pairs for flows. The models are defined as follows.

Traffic #1: For this model, the source-destination switch pair is uniformly distributed. To generate this model, a pair of different source and destination switches is randomly selected with equal probability. Then, input and output ports are randomly selected for the source and destination switches.

Traffic #2: This model simulates lightly skewed traffic. Namely, for a randomly selected source switch index i, the destination switch index k is selected from a certain range of switch indices with equal probability. The difference between k and i is kept greater than $r/4$. Specifically, k is set to a value $R(i + x)$ where

$$R(x) \triangleq x \bmod r \tag{9}$$

and $r/4 < x < 3r/4$.

Traffic #3: This model simulates heavily skewed traffic. For a randomly selected source switch index i, the destination switch index is selected from the following three numbers: $R(i + r/2 - 1)$, $R(i + r/2)$, and $R(i + r/2 + 1)$.

Traffic #4: For this model, flows are generated similarly as for Traffic #1 except that a flow passes through different second/fourth-stage switches for the source and destination sides. Thus, every flow passes through a third-stage switch. Using this rule, an equal average load is provided to a link between first/fifth-stage and second/fourth-stage switches as well as to a link between second/fourth-stage and third-stage switches.

Additionally, light traffic load was also examined for the three-stage FCN by the following model.

Traffic #5: This model is the same as Traffic #1 except that the average number of flows given to a port is 25.

The sockets were opened 2×10^6 times for Traffic #1–#4 and 5×10^5 times for Traffic #5. The metrics were measured every 1 s in the period from 401–1,900 s. The system was considered to be in equilibrium during this period. At each measurement time, the metrics were obtained from 2,304 links (1,152 uplinks and 1,152 downlinks) for the three-stage FCN, and 4,608 links for the five-stage FCN. By executing this metric calculation from 401 s to 1,900 s, 1,500 samples were obtained for one execution of the simulation program. This process was repeated 10 times with different initial values for the random function to obtain reliable results. The averages of the metrics were computed from the measured data.

Hereafter, the term *maximum* signifies the average of the sampled maxima. Thus, values labeled as the maximum are real numbers, although each sample value of the maximum metric is an integer. Similarly, the average value of the bad link metric is also a real number, although its sample is an integer.

The simulation was performed by a custom event-driven simulation program that listed events, including flow generations and flow completions, in a table. Then, the program executed the process associated with the event according to the scheduled time. Because the proposed methods were evaluated for flow characteristics, it was not necessary to consider packet behaviors or protocols that are precisely modeled by existing simulation platforms (for example, ns-3 [19]). For this purpose, the use of a custom program was more efficient. The program was built using the C language and compiled using GCC 4.8.5. The simulation was performed on a Core i3/16GB RAM PC running on CentOS 7.

C. Simulation results

Table II summarizes the simulation results for the rebalancing algorithm and the three-stage FCN fed with Traffic #1, while Table III presents the results for the balancing algorithm, the three-stage FCN, and Traffic #1.

TABLE II. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #1 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	111.701	11.154	122.816
Modification 1	111.102	7.713	66.366
Modification 2	109.276	8.710	75.086
Modifications 1 & 2	110.370	7.163	55.734

TABLE III. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #1 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	113.429	15.121	186.507
Modification 1	112.686	9.996	101.800
Modification 2	110.846	11.781	127.464
Modifications 1 & 2	112.420	9.734	97.372

Tables II and III demonstrate that the load equality was successfully improved by modifications 1 and 2. As illustrated in the tables, every metric decreased when the modifications were applied. In particular, modification 1 effectively

improved the variance and bad links metrics. Therefore, this modification is effective even though it does not affect the equality between the $D(j, k)$s. The improvements due to modification 2 were not as large as those due to modification 1. However, all metrics also decreased when modification 2 was applied. The best results were obtained for the variance and bad links metrics when both modifications 1 and 2 were applied. The improvement in the bad links metric was particularly notable. This implies that the number of flows is concentrated in a narrow range for most links. For the examined network, the bound expressed by (4) is 145 when f_0 is assumed to be 100. Thus, from Table II, the actual maximum appears much smaller than that upper bound.

In a comparison between the rebalancing and balancing algorithms, it was found that the former was always superior to the latter for any case. However, the rerouting performed by the rebalancing algorithm may cause packet reordering, which may decrease the throughput. Meanwhile, the proposed modifications considerably improved the load equality of the balancing algorithm, which does not perform rerouting. When the modifications were applied, the load equality was better for the balancing algorithm than that for the original version of the rebalancing algorithm. Therefore, a practical solution is to use the balancing algorithm with the proposed modifications.

Tables IV and V list the results for the three-stage FCN and Traffic #2, lightly skewed traffic. Table IV pertains to the case of the rebalancing algorithm, while Table V pertains to the case of the balancing algorithm.

TABLE IV. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #2 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	109.545	8.127	61.666
Modification 1	109.108	6.224	35.673
Modification 2	107.823	6.653	36.653
Modifications 1 & 2	108.570	5.865	29.412

TABLE V. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #2 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	110.703	10.365	96.802
Modification 1	110.209	7.596	55.571
Modification 2	109.038	8.521	63.876
Modifications 1 & 2	110.018	7.450	52.879

Tables IV and V reveal that every metric also decreased for the case of Traffic #2 when the modifications were applied. Similarly to the case of Traffic #1, the best result was obtained by applying both modifications 1 and 2.

Despite the results presented in Tables II–V, it cannot be concluded that the modifications are always effective for all traffic models. This is illustrated in Tables VI and VII, which display the results for Traffic #3. Table VI displays the results for the rebalancing algorithm, while Table VII displays the results for the balancing algorithm.

TABLE VI. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #3 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	105.863	4.535	12.716
Modification 1	105.762	4.383	11.224
Modification 2	105.920	4.801	15.700
Modifications 1 & 2	105.817	4.407	11.503

TABLE VII. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #3 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	106.317	4.858	15.490
Modification 1	106.157	4.633	13.310
Modification 2	106.411	5.197	19.384
Modifications 1 & 2	106.194	4.649	13.475

For Traffic #3, Tables VI and VII indicate that modification 2 is not particularly effective. As illustrated in the tables, when modification 2 was applied, every metric increased. This result can be explained by the definition of the search start index j_s used in modification 2. As seen in (8), j_s is determined by the indices of the source and destination switches. Meanwhile, a source-destination pair is selected from very few (namely, three) candidates for Traffic #3. Due to this heterogeneity in source-destination pairs, the start index j_s is not efficiently distributed over $0, 1, \ldots, m - 1$, thus leading to a less uniform load on the links.

Tables VIII and IX present the results for the five-stage FCN and Traffic #4. The results for the rebalancing algorithm are presented in Table VIII, while the results for the balancing algorithm are presented in Table IX. The tables demonstrate that each modification efficiently improved the metrics for the five-stage FCN. Similarly to the case of the three-stage FCN, the best result was obtained by applying modifications 1 and 2. However, the advantage of applying modifications 1 and 2 is not great in comparison with the case of applying only modification 1.

TABLE VIII. RESULTS FOR THE REBALANCING ALGORITHM, FIVE-STAGE FCN, AND TRAFFIC #4 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	120.116	19.729	417.707
Modification 1	119.098	13.414	283.588
Modification 2	118.053	17.403	378.419
Modifications 1 & 2	118.958	13.249	280.890

TABLE IX. RESULTS FOR THE BALANCING ALGORITHM, FIVE-STAGE FCN, AND TRAFFIC #4 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	122.126	23.903	486.419
Modification 1	120.941	15.679	319.221
Modification 2	119.295	20.266	424.823
Modifications 1 & 2	120.899	15.492	315.647

For Traffic #5, Table X shows the results for the rebalancing algorithm, and Table XI shows the results for the balancing algorithm. As shown in the tables, the characteristic for Traffic #5 differs from those for other traffic models. For the case of Traffic #5, modification 2 is more effective for the maximum and bad links metrics than modification 1. The variance metric is smaller for modification 1 than for modification 2, although the difference is almost negligible. When both modifications 1 and 2 were applied, the maximum and bad links metrics are greater than those for the case of applying modification 2. However, every metric becomes smaller by employing both modifications than that by the original version.

TABLE X. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #5 MODEL.

Algorithms	Maximum (flows)	Variance ($flows^2$)	Bad Links (links)
Original Version	36.486	11.666	122.128
Modification 1	35.885	6.495	60.932
Modification 2	33.308	6.825	37.355
Modifications 1 & 2	35.399	5.973	51.601

TABLE XI. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #5 MODEL.

Algorithms	Maximum (flows)	Variance ($flows^2$)	Bad Links (links)
Original Version	36.558	11.737	123.221
Modification 1	35.906	6.539	61.762
Modification 2	33.303	6.832	37.560
Modifications 1 & 2	35.498	5.995	51.809

As illustrated in Tables I–XI, the effectiveness of each technique depends on the traffic model. For Traffic #1–#4, modification 1 is more effective. For Traffic #3, modification 2 does not improve the metrics. However, for Traffic #5, modification 2 works very well. When both modifications 1 and 2 were applied for Traffic #1 and #2, most metric values became smaller than those for the case of applying either one of modification 1 or 2. However, even if both modifications were applied, the metric values almost equaled those for the case of applying modification 1 for Traffic #3. Furthermore, when both modifications were applied for Traffic #5, some metric values became greater than those for the case of applying modification 2.

These results suggest that the best performance is obtained by selecting the technique depending on the characteristic of traffic load. For heavy and skewed loads, modification 1 should be used. If the load is light, modification 2 will be a better choice. However, if predicting the load characteristic is difficult, both modifications 1 and 2 should be applied. By applying both modifications, the metrics were considerably improved compared to the original version for every examined traffic model. Thus, applying both modifications is an effective way to improve load equality, although it does not always yield the best results.

D. Discussion

In Section VI.C, the proposed techniques are evaluated through the number of flows that pass through a link. However, it is uncertain whether this advantage in flow number equality directly leads to the improvement of the performance experienced by users. For the clarification of this point, it is necessary to perform additional computer simulation, which precisely models the packet-level behaviors including protocol and queueing processes. Through this simulation, the performance improvement will be confirmed through the metrics such as throughputs and response time, which are experienced by users. Actually, the packet level simulation of the flow diffusion algorithms has been partly done and reported in [32]. In [32], the TCP throughput is measured for the bulk data transfer application. The result shows that the number of flows with small throughputs successfully decreases through equal flow diffusion obtained by the balancing algorithm. This characteristic implies that the proposed techniques will effectively reduce the probability of throughput degradation because the techniques successfully improve the flow number equality.

As a future study, implementation of the rebalancing or balancing algorithm with the proposed techniques may be required to assess the feasibility and advantage of the approach. For implementation, it will be necessary to employ a mechanism that enables flow-based routing, for example, OpenFlow [33]. Additionally, the start and end of a flow must be detected to run the rebalancing algorithm. This detection of flow start/end will be achieved by the techniques presented in [34]. However, further study is necessary to clarify if it is possible to set the routing table entry from a flow detection in a practical processing time.

VII. CONCLUSION AND FUTURE WORK

This paper investigates two techniques to improve the rebalancing algorithm [12], which diffuses flows in an FCN. The first technique decreases the difference between the uplink loads by adding a criterion for determining the middle switch used in the routing or rerouting processes. In addition, the load equality depends on the scanning order of the middle switch indices. Based on this, the second technique determines the start index for scanning to balance the loads. The two techniques were applied to the rebalancing and balancing algorithms and were evaluated using computer simulations. The balancing algorithm is a version of the rebalancing algorithm that is modified to omit the rerouting process. The results demonstrated that the proposed techniques successfully improved load equality.

By expanding the work of a previous study [1], this study examined the application of the techniques to a five-stage FCN. For a five-stage FCN, the upper bound was analyzed for the number of flows on a link on the basis of the upper bound for the three-stage FCN case. To the best of the author's knowledge, this bound has never been reported in the literature. Thus, the derivation of that bound is an important contribution. In addition, computer simulations confirmed that the proposed techniques were effective for the five-stage

FCN case. This effectiveness has also not been reported in previous studies.

As another expansion of [1], performance against a broader range of traffic models was tested via computer simulations. Several of the models employed skewed traffic matrices, for which the source-destination pair of a generated flow was selected from a limited number of candidates. In addition, one model simulated a lighter traffic load than other models. The results demonstrated that the second technique was not effective for a highly skewed traffic matrix. However, the second technique is more effective for light traffic loads. In addition, load equality was improved for every tested traffic model when both proposed techniques were applied. Thus, as an important result, it was found that both techniques should be used independent of traffic loads.

Further study is necessary to determine how the load equality enhanced by the proposed techniques affects packet-level performance, such as packet latency and throughput. To achieve this, a more precise packet-level computer simulation is required. The implementation of the two proposed techniques is also important for future work.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP19K11928. The author would like to thank Enago (www.enago.jp) for the English language review.

REFERENCES

[1] S. Ohta, "Techniques to improve a flow diffusion algorithm for folded Clos networks," The Eighteenth International Conference on Networks (ICN 2019), Valencia, Spain, Mar. 2019, pp. 68–73, ISBN: 978-1-61208-695-8.

[2] A. Singh et al., "Jupiter rising: a decade of Clos topologies and centralized control in Google's datacenter network," The 2015 ACM Conference on Special Interest Group on Data Communication, London, United Kingdom, Aug. 2015, pp. 183–197, ISBN: 978-1-4503-3542-3, doi: 10.1145/2785956.2787508.

[3] Z. Guo and Y. Yang, "On nonblocking multicast fat-tree data center networks with server redundancy," IEEE Trans. on Computers, vol. 64, no. 4, pp. 1058–1073, Apr. 2014.

[4] C. Guo et al., "DCell: a scalable and fault-tolerant network structure for data centers," ACM SIGCOMM 2008 Conference on Data communication, Seattle, WA, USA, Aug. 2008, pp. 75–86, ISBN: 978-1-60558-175-0, doi: 10.1145/1402958.1402968.

[5] C. Guo et al., "BCube: a high performance, server-centric network architecture for modular data centers," ACM SIGCOMM 2009 Conference on Data communication, Barcelona, Spain, Aug. 2009, pp. 63–74, ISBN: 978-1-60558-594-9, doi:10.1145/1592568.1592577.

[6] N. Farrington and A. Andreyev, "Facebook's data center network architecture," 2013 Optical Interconnects Conference, Santa Fe, NM, USA, May 2013, pp. 49–50, ISSN: 2376-8665, doi: 10.1109/OIC.2013.6552917.

[7] A. Greenberg et al., "VL2: a scalable and flexible data center network," Communications of the ACM, vol. 54, no. 3, pp. 95–104, Mar. 2011.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," ACM SIGCOMM 2008 conference on Data communication, Seattle, WA, USA, Aug. 2008, pp. 63–74, ISBN: 978-1-60558-175-0, doi:10.1145/1402958.1402967.

[9] C. Clos, "A study of nonblocking switching networks," Bell System Technical Journal, vol. 32, no. 2, pp. 406–424, Mar. 1953.

[10] S. Scott, D. Abts, J. Kim, and W.J. Dally, "The BlackWidow high-radix Clos network," The 33rd Annual International Symposium on Computer Architecture (ISCA '06), Boston, MA, USA, June 2006, pp. 16–28, ISBN:0-7695-2608-X, ISSN: 1063-6897, doi: 10.1109/ISCA.2006.40.

[11] E. Zahavi, I. Keslassy, and A. Kolodny, "Distributed adaptive routing for big-data applications running on data center networks," 2012 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '12), Austin, Tx, USA, Oct. 2012, pp. 99–110, ISBN: 978-1-4503-1685-9.

[12] S. Ohta, "Flow diffusion algorithms based on local and semi-local information for folded Clos networks," The Fourth International Conference on Electronics and Software Science (ICESS 2018), Takamatsu, Japan, Nov. 2018, pp. 46–54, ISBN: 978-1-941968-52-9.

[13] A.Jajszczyk, "Nonblocking, repackable, and rearrangeable Clos networks: fifty years of theory evolution," IEEE Communications Magazine, vol. 41, no. 10, pp.28–33, Oct. 2003.

[14] D. G. Cantor, "On nonblocking switching networks," Networks, vol. 1, no. 4, pp. 367–377, winter 1971.

[15] H.-A. Kim and D. R. O'Hallaron, "Counting network flows in real time," IEEE Global Telecommunications Conference (GLOBECOM 2003), San Francisco, CA, USA, Dec. 2003, pp. 3888–3893, ISBN: 0-7803-7974-8, doi: 10.1109/GLOCOM.2003.1258959.

[16] J. W. Roberts, "Traffic theory and the Internet," IEEE Communications Magazine, vol. 39, no. 1, pp.94–99, Jan. 2001.

[17] L. A. Bassalygo, I. I. Grushko, and V. I. Neiman, "The Structures of One-Sided Connecting Networks," The Sixth International Teletraffic Congress (ITC 6), Munich, Germany, Sept. 1970. Available from https://itc-conference.org/en/itc-library/itc6.html, 2019.11.18.

[18] G. Broomel and J. R. Heath, "Classification categories and historical development of circuit switching topologies," Computing Surveys, vol. 15, no. 2, pp. 95–133, June 1983.

[19] N. Fujii, "Application of a rearrangement algorithm for digital cross-connect system control," The Eighth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '89), Ottawa, Canada, Apr. 1989, pp. 228–233, ISBN: 0-8186-1920-1, doi: 10.1109/INFCOM.1989.101458.

[20] M. K. Panda, T. Venkatesh, V. Sridhar, and Y. N. Singh, "Architecture for a class of scalable optical cross-connects," First International Conference on Broadband Networks, San Jose, CA, USA, Oct. 2004, pp. 233–242, ISBN: 0-7695-2221-1, doi: 10.1109/BROADNETS.2004.16.

[21] Y. –K. Chen and C. –C. Lee, "Fiber Bragg grating-based large nonblocking multiwavelength cross-connects," Journal of Lightwave Technology, vol. 16, no 10, pp. 1746–1756, Oct. 1998.

[22] Y. –H. Kao, N. Alfaraj, M. Yang and H. J. Chao, "Design of high-radix Clos network-on-chip," The 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS '10), Grenoble, France, May 2010, pp. 181–188, ISBN: 978-0-7695-4053-5 doi: 10.1109/NOCS.2010.27.

[23] A. Zia, S. Kannan, G. Rose and H. J. Chao, "Highly-scalable 3D Clos NOC for many-core CMPs," The 8th IEEE International NEWCAS Conference 2010 (NEWCAS2010), Montreal, QC, Canada, June 2010, pp. 229–232, doi: 10.1109/NEWCAS.2010.5603776.

[24] A. Joshi et al., "Silicon-photonic Clos networks for global on-chip communication," The 2009 Third ACM/IEEE International Symposium on Networks-on-Chip (NOCS '09), , San Diego, CA, USA, May 2009, pp. 124 – 133, ISBN: 978-1-4244-4142-6, doi: 10.1109/NOCS.2009.5071460.

[25] F. Hassen and L. Mhamdi, "A Clos-network switch architecture based on partially buffered crossbar fabrics," 2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, USA, Aug. 2016, pp. 45–52, ISSN: 2332-5569, doi: 10.1109/HOTI.2016.020.

[26] F. Hassen and L. Mhamdi, "High-capacity Clos-network switch for data center networks," 2017 IEEE International Conference on Communications (ICC 2017), Paris, France, May 2017, paper NGNI07-1, pp. 1–7, ISSN: 1938-1883, ISBN: 978-1-4673-8999-0, doi: 10.1109/ICC.2017.7997147.

[27] S. Yang, S. Xin, Z. Zhao, and B. Wu, "Minimizing packet delay via load balancing in Clos switching networks for datacenters," 2016 International Conference on Networking and Network Applications (NaNA 2016), Hakodate, Japan, July 2016, pp. 23–28, doi: 10.1109/NaNA.2016.14.

[28] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Micro load balancing in data centers with DRILL," The 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV), Philadelphia, PA, USA, Nov. 2015, paper 17, ISBN: 978-1-4503-4047-2, doi:10.1145/2834050.2834107.

[29] L. G. Valiant, "A scheme for fast parallel communication," SIAM J. Computing, vol. 11, no. 2, pp. 350–361, May 1982.

[30] S. Ohta, "A simple control algorithm for rearrangeable switching networks with time division multiplexed links," IEEE J. on Selected Areas in Communications, vol. SAC-5, no. 8, pp.1302–1308, Oct. 1987.

[31] ns developers, *ns-3, Network Simulator* [Online], Available from https://www.nsnam.org/, 2019.11.09.

[32] S. Ohta, "TCP throughput achieved by a folded Clos network controlled by different flow diffusion algorithms," International Journal of Information and Electronics Engineering, in press..

[33] O. Cocker and S. Azodolmolky, Software-Defined Networking with OpenFlow - Second Edition: Deliver innovative business solutions, Packt, Birmingham, UK, 2017.

[34] S. Zhu and S. Ohta, "Real-time flow counting in IP networks: strict analysis and design issues," Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications, vol. 2, no. 2, pp. 7–17, Feb. 2012.

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