ICNS 2018

The Fourteenth International Conference on Networking and Services

ISBN: 978-1-61208-633-0

May 20 - 24, 2018

Nice, France

ICNS 2018 Editors

Kevin Daimi, University of Detroit Mercy, USA
Eugen Borcoci, University "Politehnica" of Bucharest (UPB), Romania

ICNS 2018

Foreword

The Fourteenth International Conference on Networking and Services (ICNS 2018), held between May 20 - 24, 2018 - Nice, France, continued a series of events targeting general networking and services aspects in multi-technologies environments. The conference covered fundamentals on networking and services, and highlighted new challenging industrial and research topics. Network control and management, multi-technology service deployment and assurance, next generation networks and ubiquitous services, emergency services and disaster recovery and emerging network communications and technologies were considered.

IPv6, the Next Generation of the Internet Protocol, has seen over the past three years tremendous activity related to its development, implementation and deployment. Its importance is unequivocally recognized by research organizations, businesses and governments worldwide. To maintain global competitiveness, governments are mandating, encouraging or actively supporting the adoption of IPv6 to prepare their respective economies for the future communication infrastructures. In the United States, government's plans to migrate to IPv6 has stimulated significant interest in the technology and accelerated the adoption process. Business organizations are also increasingly mindful of the IPv4 address space depletion and see within IPv6 a way to solve pressing technical problems. At the same time IPv6 technology continues to evolve beyond IPv4 capabilities. Communications equipment manufacturers and applications developers are actively integrating IPv6 in their products based on market demands.

IPv6 creates opportunities for new and more scalable IP based services while representing a fertile and growing area of research and technology innovation. The efforts of successful research projects, progressive service providers deploying IPv6 services and enterprises led to a significant body of knowledge and expertise. It is the goal of this workshop to facilitate the dissemination and exchange of technology and deployment related information, to provide a forum where academia and industry can share ideas and experiences in this field that could accelerate the adoption of IPv6. The workshop brings together IPv6 research and deployment experts that will share their work. The audience will hear the latest technological updates and will be provided with examples of successful IPv6 deployments; it will be offered an opportunity to learn what to expect from IPv6 and how to prepare for it.

Packet Dynamics refers broadly to measurements, theory and/or models that describe the time evolution and the associated attributes of packets, flows or streams of packets in a network. Factors impacting packet dynamics include cross traffic, architectures of intermediate nodes (e.g., routers, gateways, and firewalls), complex interaction of hardware resources and protocols at various levels, as well as implementations that often involve competing and conflicting requirements.

Parameters such as packet reordering, delay, jitter and loss that characterize the delivery of packet streams are at times highly correlated. Load-balancing at an intermediate node may, for example, result in out-of-order arrivals and excessive jitter, and network congestion may manifest as packet losses or large jitter. Out-of-order arrivals, losses, and jitter in turn may lead to unnecessary retransmissions in TCP or loss of voice quality in VoIP.

With the growth of the Internet in size, speed and traffic volume, understanding the impact of underlying network resources and protocols on packet delivery and application performance has assumed a critical importance. Measurements and models explaining the variation and interdependence of delivery characteristics are crucial not only for efficient operation of networks and network diagnosis, but also for developing solutions for future networks.

Local and global scheduling and heavy resource sharing are main features carried by Grid networks. Grids offer a uniform interface to a distributed collection of heterogeneous computational, storage and network resources. Most current operational Grids are dedicated to a limited set of computationally and/or data intensive scientific problems.

Optical burst switching enables these features while offering the necessary network flexibility demanded by future Grid applications. Currently ongoing research and achievements refers to high performance and computability in Grid networks. However, the communication and computation mechanisms for Grid applications require further development, deployment and validation.

We take here the opportunity to warmly thank all the members of the ICNS 2018 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ICNS 2018.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ICNS 2018 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICNS 2018 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the fields of networking and services.

We are convinced that the participants found the event useful and communications very open. We also hope that Nice provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

ICNS 2018 Chairs:

ICNS Steering Committee

Eugen Borcoci, University "Politehnica" of Bucharest (UPB), Romania
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Mary Luz Mouronte López, Universidad Francisco de Vitoria - Madrid, Spain
Massimo Villari, Universita' di Messina, Italy
Éric Renault, Institut Mines-Télécom - Télécom SudParis, France
Robert Bestak, Czech Technical University in Prague, Czech Republic
Young-Joo Suh, POSTECH (Pohang University of Science and Technology), Korea
Gledson Elias, Federal University of Paraíba (UFPB), Brazil
Rui L.A. Aguiar, University of Aveiro, Portugal
Ivan Ganchev, University of Limerick, Ireland / Plovdiv University "Paisii Hilendarski", Bulgaria

ICNS Industry/Research Advisory Committee

Steffen Fries, Siemens, Germany
Alex Sim, Lawrence Berkeley National Laboratory, USA
Lorenzo Mossucca, Istituto Superiore Mario Boella, Italy
Jeff Sedayao, Intel Corporation, USA
Juraj Giertl, T-Systems, Slovakia

ICNS 2018

ICNS Steering Committee
Eugen Borcoci, University "Politehnica" of Bucharest (UPB), Romania
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Mary Luz Mouronte López, Universidad Francisco de Vitoria - Madrid, Spain
Massimo Villari, Universita' di Messina, Italy
Éric Renault, Institut Mines-Télécom - Télécom SudParis, France
Robert Bestak, Czech Technical University in Prague, Czech Republic
Young-Joo Suh, POSTECH (Pohang University of Science and Technology), Korea
Gledson Elias, Federal University of Paraíba (UFPB), Brazil
Rui L.A. Aguiar, University of Aveiro, Portugal
Ivan Ganchev, University of Limerick, Ireland / Plovdiv University "Paisii Hilendarski", Bulgaria

ICNS Industry/Research Advisory Committee
Steffen Fries, Siemens, Germany
Alex Sim, Lawrence Berkeley National Laboratory, USA
Lorenzo Mossucca, Istituto Superiore Mario Boella, Italy
Jeff Sedayao, Intel Corporation, USA
Juraj Giertl, T-Systems, Slovakia

ICNS 2018 Technical Program Committee

Wessam Afifi, Mavenir Systems in Richardson, USA
Rui L.A. Aguiar, University of Aveiro, Portugal
Pouyan Ahmadi, George Mason University, USA
Mehmet Aksit, University of Twente, Netherlands
Markus Aleksy, ABB AG, Germany
Alexandros Apostolos Boulogeorgos, Aristotle University of Thessaloniki, Greece
Patrick Appiah-Kubi, University of Maryland University College, USA
Mohammad M. Banat, Jordan University of Science and Technology, Jordan
Meriem Kassar Ben Jemaa, National Engineering School of Tunis (ENIT), Tunisia
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Jihen Bennaceur, École nationale des sciences de l'informatique, Tunisia
Luis Bernardo, Universidade Nova de Lisboa, Portugal
Robert Bestak, Czech Technical University in Prague, Czech Republic
Ateet Bhalla, Independent Consultant, India
Eugen Borcoci, University "Politehnica" of Bucharest (UPB), Romania
Fernando Boronat Seguí, Universidad Politécnica De Valencia-Campus De Gandia, Spain
Safdar Hussain Bouk, Kyungpook National University, Daegu, Republic of Korea
Christos Bouras, University of Patras / Computer Technology Institute and Press - Diophantus, Greece
An Braeken, Vrije Universiteit Brussels (VUB), Belgium
Maria-Dolores Cano, Universidad Politécnica de Cartagena, Spain
José Cecílio, University of Coimbra, Portugal
Jin-Hee Cho, U.S. Army Research Laboratory (USARL), USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Performance Analysis of a Full Duplex MAC Protocol with Binary Exponential Backoff

Sammy Chan

Department of Electronic Engineering
City University of Hong Kong
Hong Kong SAR
P.R. China
Email: eeschan@cityu.edu.hk

***Abstract*—The full duplex mode in wireless communication allows simultaneous data transmission and reception on a single channel, hence increasing the data rate and reducing the access delay. In this paper, we propose a TDMA-based medium access control protocol to realize the benefit of full duplex communication. This protocol deploys the binary exponential backoff mechanism to resolve contention. We develop an analytical model to evaluate its performance. This protocol is suitable for future generations of mobile telecommunication networks, such as 5G networks.**

Keywords–Full duplex; MAC; Exponential backoff.

I. INTRODUCTION

The next generation wireless networks, such as the fifth generation mobile telecommunications networks (5G networks) and the new generation of Wireless Local Area Networks (WLANs), need to support higher user density. This is due to the tremendous success of smart phones and increasing deployment of the Internet of Things (IoT) technology. Moreover, users are running more and more bandwidth intensive and time-sensitive multimedia applications. These two factors drive the next generation wireless networks to provide much higher data rates. For example, 5G networks target to provide a data rate of several orders of magnitude higher than that of 4G networks.

In order to enable the next generation wireless networks to deliver a tremendously increased data rate, the spectrum efficiency needs to be significantly enhanced. This can be achieved by a multitude of new technologies, such as spectrum efficiency optimization, cooperative communications, multiple-input multiple-output (MIMO) [1], non-orthogonal multiple access [2] [3] and full duplex (FD) transmission [4] [5].

In currently deployed wireless networks, nodes cannot transmit and receive on the same frequency band at the same time. Either frequency-division duplexing (FDD) or time-division duplexing (TDD) transmission techniques are commonly deployed. Both of them operate in half-duplex modes. In FDD, two frequency channels are needed to support bidirectional communications; one for uplink and one for downlink. In TDD mode, the uplink and downlink data are sent in orthogonal time-slots. On the other hand, FD transmission allows a node to simultaneously send and receive data on a single channel.

In the approach proposed in [4], FD transmission is realized by the use of analog and digital cancellation. It requires one receiving (RX) antenna, one transmitting (TX) antenna and a balanced/unbalunced (Balun) transformer to be installed at each node. First, analog cancellation is used. The signal from the TX antenna is cancelled at the RX antenna by the inverted signal generated by the Balun transformer. It can cancel a minimum of 45 dB across a 40 MHz spectrum. Then, digital cancellation is employed to further reduce self-interference by up to 73 dB for 10 MHz OFDM signal. In [5], a design for a wideband multiple-antenna self-interference canceller for orthogonal frequency-division-multiplexing (OFDM) systems is presented. It combines three methods to reduce the self-interference. First, multiple antennas are suitably placed on a node for passive suppression. It maximizes the attenuation of the self-interference signal over the channel between the transmitter and receiver antennas of the same node. Second, a per-subcarrier per-receive-antenna analog self-interference canceler is used. Third, a digital self-interference canceller in baseband is implemented. The extensive experimental results demonstrate that this three-stage design achieves a minimum and maximum cancellation of 70 dB and 100 dB, respectively, with a median of 85 dB.

When FD transmission is enabled, it can be deployed in two different modes. As shown in Figure 1(a) , both nodes 1 and 2 are equipped with one TX antenna and one RX antenna. They can send and receive data simultaneously over the single channel. This mode is referred to as *FD-bidirectional transmission*. Alternatively, as shown in Figure 1(b), node 1 is sending data to node 2. While node 2 is receiving the data, it can immediately forward it to node 3. In other words, node 2 is simultaneously receiving and transmitting. This mode is referred to as *FD-relay transmission*. For this mode, only node 2 needs to have two antennas installed; one TX antenna and one RX antenna. Since node 1 and node 3 do not transmit and receive simultaneously, each of them only need to have one antenna.

Having FD transmission enabled at the physical layer, there is a need for a suitable medium access control (MAC) protocol to realize its benefit. In [6], a FD MAC protocol is proposed for nodes equipped with directional antennas to improve the throughput of a multi-hop flow using the FD-relay mode. In [7], a unified MAC protocol is proposed to establish either FD-bidirectional or FD-relay transmission. The protocol is based on the RTS/CTS mechanism. In [8]-[10], MAC protocols are proposed for the case that only the base station has FD capability while user nodes still operate in half duplex mode. In [11] [12], FD MAC protocols are proposed for the more general case that both base station and user nodes have FD capability. In [13], the authors propose a CSMA-based MAC

 ISBN: 978-1-61208-633-0

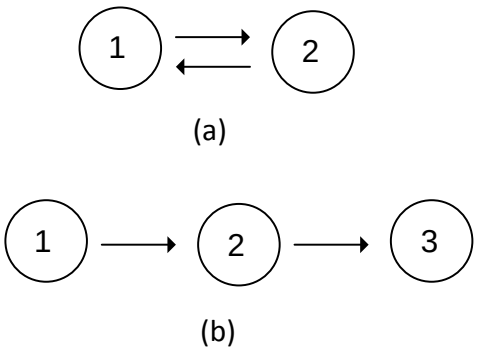


Figure 1. FD transmission modes.

protocol in which transmitters use FD techniques to monitor the channel usage during transmitting, and backoff as soon as collision is observed.

So far, MAC protocols for FD transmission are mainly designed for WLANs. In this paper, we propose a MAC protocol for cellular networks in which both base stations and user equipment (UE) can support FD transmission. The proposed protocol is contention based and requires UE to use the binary exponential backoff (BEB) algorithm for contention resolution. The remainder of this paper is organized as follows. In Section II, we present our proposed MAC protocol. Then, in Section III, we develop an analytical model to characterize the average packet delay and its standard deviation under the saturated condition. In Section IV, we verify the accuracy of our model by comparing with simulation results. Finally, Section V concludes the paper.

II. MAC PROTOCOL

The frame structure for the proposed MAC protocol is shown in Figure 2. In this protocol, the channel is time slotted and organized into frames. Each frame has a duration of $\mathcal{E}$, and consists of a request subframe, a full duplex data subframe, an information subframe and a downlink data subframe, with duration T_r, T_f, T_i and T_h, respectively.

A request subframe has m request slots, each of length t. In order to gain the right to send a data packet, an UE first needs to send a bandwidth request to a randomly chosen slot in the request subframe. If there is only one request submitted to the request slot, the request is successful. On the other hand, if there are two or more UE submitting their requests in the same request slot, collision occurs.

In a full duplex data subframe, there are d data slots. Upon receiving the bandwidth requests, the BS will announce the contention result in the full duplex map (FD Map) of the information subframe. It assigns a data slot in the full duplex data subframe of the next frame to each successful UE. Each data slot is of length T ($T \gg t$), which is the transmission time of a data packet (all packets are assumed to have the same length throughout this paper). Here, we assume that a data slot is randomly chosen and assigned to a successful UE. Since full duplex transmission is supported, when an UE is transmitting a packet to the BS in the assigned data slot, the BS can simultaneously send downlink traffic to the corresponding UE. For those unsuccessful UE, they need to execute the BEB algorithm to resolve contention. The complete process of sending request is described as below.

Initially, before sending a request each UE starts its own backoff process by randomly selecting a backoff time in the range $[0, CW-1]$, where CW is the contention window. Here the backoff time represents the number of request slots that must pass before the request can be submitted. At the first attempt, CW is set equal to W, the minimum contention window. If the request is unsuccessful, then the contention window size is multiplied by $\lambda = 2$, and another backoff period is initiated. This process is repeated for each subsequent request failure, i.e., $W_i = \lambda^i W$, where i is the number of re-attempts. Window doubling continues until the maximum possible value, $CW_{max} = \lambda^r W, r \geq 1$, is reached. If the request is unsuccessful after r attempts, the window is maintained at CW_{max} for the remaining attempts until the request is successful, or until the maximum number of attempts, $R \geq r$, is reached. If the request is still unsuccessful after R attempts, the packet is discarded.

In Figure 2, we show an example in which an UE sends a request in frame 1 but fails. It carries out the BEB process and finally can have a duplex data transmission in frame n. If the traffic between UE and the BS is symmetric, there is no need to provision downlink data subframes. However, to cater for the case in which some UE only have downlink traffic, the downlink data subframe with b data slots is provisioned. When an UE only has downlink traffic, the BS will inform it in the downlink map (DL Map) of the information subframe to receive a packet in a particular data slot in the downlink data subframe.

III. ANALYTICAL MODEL

Consider a cell with N UE. Assume that all UE are in saturated condition, i.e., they always have packets to send. The packet delay X is defined as the time duration from its first bandwidth request until the packet transmission has finished. Note that if the bandwidth reservation of a packet is successful, the packet will be removed from the head of the queue into a temporary buffer and transmitted to the channel in the coming full duplex data subframe. Thus, the bandwidth request of the next packet (now become the head of the queue) can be sent in the request subframe of the next frame.

A. Unsuccessful Request Probability

Let p be the probability that a request sent by an UE is unsuccessful, which is given by

$$p = 1 - (1 - p_c)(1 - p_d), \tag{1}$$

where p_c and p_d are probabilities that a request sent by an UE is unsuccessful due to collision with other requests, or lack of data slots in the full duplex data subframe of the next frame, respectively. In the following, we show that both p_c and p_d can be expressed as a function of the probability p, hence fixed point equations can be established to calculate the individual probabilities.

A request is successfully transmitted on the first attempt with probability $1 - p$ (ignoring a normalisation factor that we will introduce later). Recall that the contention window is initially set to W, the average number of elapsed backoff slots

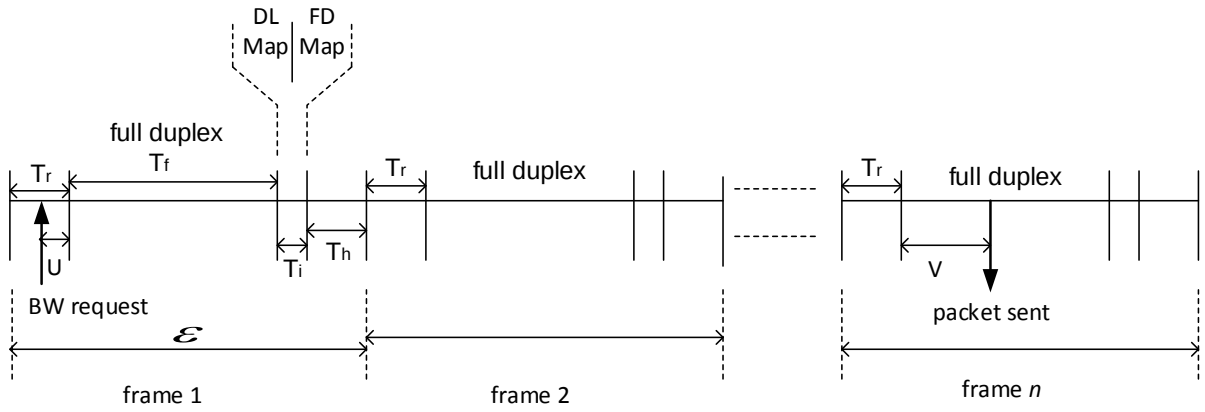


Figure 2. Frame structure of the proposed MAC protocol.

before such a request is $\frac{m}{2} + (W-1)/2$. The first term is due to the fact that an UE cannot start a new backoff period for its next request immediately after the previous one in the same frame but has to wait until the next frame. And because requests are uniformly chosen among the m request slots in each frame, the average number of backoff slots wasted until the next frame is $\frac{m}{2}$. The second term represents the average number of backoff slots an UE has to wait before attempting to send a request according to the BEB mechanism described in Section II.

If the first transmission fails, the request is successfully transmitted on the second attempt with probability $p(1-p)$. The average number of elapsed backoff slots in this case is $\frac{m}{2} + (\lambda W - 1)/2$. Continuing this argument until the R^{th} attempt yields the overall average number of elapsed backoff slots before a successful request transmission:

$$\begin{aligned} B_{avg} &= \frac{m}{2} + \eta \sum_{i=0}^{r-1} p^i \left(\frac{\lambda^i W - 1}{2}\right) + \\ &\quad \eta \left(\frac{\lambda^r W - 1}{2}\right) \sum_{i=r}^{R-1} p^i \\ &= \frac{m}{2} + \frac{\eta W (1 - (\lambda p)^r)}{2(1-\lambda p)} - \\ &\quad \frac{1-p^r}{2(1-p^R)} + \frac{(\lambda^r W - 1)(p^r - p^R)}{2(1-p^R)}, \end{aligned} \tag{2}$$

where

$$\eta = (1-p)(1-p^R)^{-1},$$

and $(1-p^R)$ is a normalisation factor.

Note that assuming a request will be eventually successful, then B_{avg} is the average number of backoff slots an UE has to wait before sending requests, i.e., it is an average inter-arrival time of requests in this system. Therefore, the probability that an UE attempts to send the request in a slot is given by

$$\tau = 1/(B_{avg} + 1).$$

Since there are N saturated UE, the probability p_c that a request sent by an UE is unsuccessful due to collision with other requests can be expressed as

$$p_c = 1 - (1-\tau)^{N-1}. \tag{3}$$

The probability that there are $j,\ 0 \le j \le k = \min(m, N)$ successful requests among m request slots can be approximated based on a truncated binomial distribution

$$Q(j) = \frac{\binom{m}{j} \xi^j (1-\xi)^{m-j}}{\sum_{i=0}^{k} \binom{m}{i} \xi^i (1-\xi)^{m-i}}, \tag{4}$$

where $\xi = N\tau(1-\tau)^{N-1}$ is the probability that a request sent in a request slot will be successful given that there are N UE each attempting to send requests with probability τ.

The probability that a request is unsuccessful due to lack of data slots in the subsequent frame can then be expressed as

$$p_d = \frac{\sum_{j=d+1}^{k} (j-d) Q(j)}{\sum_{j=0}^{k} j Q(j)}, \tag{5}$$

recalling that d is the number of data slots set by BS in the full duplex data subframe. Equations (1), (3) and (5) create a fixed point formulation from which p can be computed numerically. Note that such a fixed point analysis is used in [14] to model the performance of the IEEE 802.11 distributed coordination function.

B. Delay Model

Consider a tagged UE. Let U be a random variable (RV) representing the time duration from the time the UE sends a request until the end of the request subframe. Assume that after certain number of attempts, the request is successful and a data slot is assigned in a full duplex data subframe. Let V be a RV representing the time duration between the beginning of that subframe and the end of the assigned data slot.

Given that the tagged UE is successful in its first attempt of sending a request, the packet delay X is therefore given by

$$X^{(0)} = U + \mathcal{E} + V \quad \text{w.p.} \quad 1-p,$$

where $\mathcal{E}$ is the duration of a frame and equal to $T_r + T_f + T_i + T_h$, $T_r = mt$, $T_f = dT$, and w.p. stands for "with probability".

For the case when the tagged UE is not successful in its first attempt but is successful in the second attempt of sending request, the service time can be expressed as

$$X^{(1)} = U + Y^{(1)} + V \quad \text{w.p.} \quad p(1-p),$$

where $Y^{(1)}$ is a random sum of a frame duration $\mathcal{E}$, and $p(1-p)$ is the probability that the request is successful in the second attempt. The variable $Y^{(1)}$ is originated from the fact that an UE will have to wait for a random backoff period before sending its request which is uniformly chosen from the new contention window. In this second attempt of sending request, an UE will choose its backoff time uniformly in $[0, W_1 - 1]$; $W_1 = \lambda W$ and $Y^{(1)}$ can be calculated as

$$Y^{(1)} = \sum_{i=0}^{1} K^{(i)} \mathcal{E},$$

where $K^{(i)}$ is a discrete random variable with the following distribution:

$$K^{(i)} = \begin{cases} 1 & \text{w.p.} \quad m/W_i, \\ 2 & \text{w.p.} \quad m/W_i, \\ & \ldots \\ A_i - 1 & \text{w.p.} \quad m/W_i, \\ A_i & \text{w.p.} \quad 1 - \frac{(A_i-1)m}{W_i}, \end{cases} \tag{6}$$

where $A_i = \lceil W_i/m \rceil$, $i = 1, .., R-1$. The $\lceil z \rceil$ operator gives a minimum integer value that is greater or equal to z. For $i = 0$, we define $K^{(0)} = 1$ w.p. one. Note that if $A_i = 1$, i.e., $m \geq W_i$, then $K^{(i)} = 1$ with probability one.

In general, the packet delay X can be expressed as

$$X = X^{(i)} \quad \text{w.p.} \quad \eta p^i, \ 0 \leq i \leq R-1, \tag{7}$$

where

$$X^{(i)} = U + Y^{(i)} + V, \tag{8}$$

and

$$Y^{(i)} = \sum_{j=0}^{i} K^{(j)} \mathcal{E}.$$

To complete the expression of X, we now determine the probability mass function (pmf) of U and V. As the tagged UE uniformly chooses the backoff before sending a request, the pmf of the U can be approximated as below

$$U = \begin{cases} mt & \text{w.p.} \quad 1/m, \\ (m-1)t & \text{w.p.} \quad 1/m, \\ \vdots & \\ t & \text{w.p.} \quad 1/m. \end{cases} \tag{9}$$

As the BS uniformly allocates data slots among successful requests, the pmf of V can be expressed as

$$V = \begin{cases} T & \text{w.p.} \quad \sum_{j=0}^{k'-1} \frac{1}{j+1} \ q(j), \\ 2T & \text{w.p.} \quad \sum_{j=1}^{k'-1} \frac{1}{j+1} \ q(j), \\ \vdots & \\ k'T & \text{w.p.} \quad \frac{1}{k'} \ q(k'-1), \end{cases} \tag{10}$$

where $q(j)$ is the probability that there are $j \geq 0$ successful requests other than the tagged UE in a frame. The probability $q(j)$ follows a truncated binomial distribution

$$q(j) = Q(j+1)/(1 - Q(0)), \ 0 \leq j \leq k-1, \tag{11}$$

where $k = \min(m, N)$ and $k' = \min(k, d)$ and $Q(j)$ is given in (4).

From (7), we obtain

$$\mathrm{E}[X] = \eta \sum_{i=0}^{R-1} p^i \, \mathrm{E}[X^{(i)}], \tag{12}$$

$$\mathrm{Var}[X] = \eta \sum_{i=0}^{R-1} p^i (\mathrm{Var}[X^{(i)}] + (\mathrm{E}[X^{(i)}] - \mathrm{E}[X])^2).$$

The mean and variance of $X^{(i)}$ are derived from (8) as

$$\begin{aligned} \mathrm{E}[X_i] &= E[U] + E[Y^{(i)}] + E[V], \\ \mathrm{Var}[X_i] &= \mathrm{Var}[U] + \mathrm{Var}[Y^{(i)}] + \mathrm{Var}[V], \end{aligned} \tag{13}$$

where

$$\mathrm{E}[Y_i] = \mathcal{E} \sum_{j=0}^{i} E[K^{(j)}],$$

$$\mathrm{Var}[Y_i] = \mathcal{E}^2 \sum_{j=0}^{i} \mathrm{Var}[K^{(j)}].$$

From (6), it can be shown that

$$E[K^{(j)}] = \begin{cases} 1 & j = 0, \\ A_j - A_j(A_j - 1)\frac{m}{2\lambda^j W} & j = 1, .., r-1, \\ A_j - A_j(A_j - 1)\frac{m}{2\lambda^r W} & j = r, .., R-1, \end{cases} \tag{14}$$

and

$$\mathrm{Var}[K^{(j)}] = \overline{K^{(j)^2}} - (E[K^{(j)}])^2,$$

where $\overline{K^{(j)^2}}$ is the second moment of $K^{(j)}$ and is given by

$$\overline{K^{(j)^2}} = \begin{cases} 1 & j = 0, \\ A_j^2 - A_j(A_j - 1)(1 + 4A_j)\frac{m}{6\lambda^j W} & j = 0, 1, .., r-1, \\ A_j^2 - A_j(A_j - 1)(1 + 4A_j)\frac{m}{6\lambda^r W} & j = r, .., R-1. \end{cases}$$

It remains to determine $E[U]$, $\mathrm{Var}[U]$, $E[V]$ and $\mathrm{Var}[V]$ from (9) and (10), which can be expressed as

$$\begin{aligned} E[U] &= (m+1)t/2, \\ \mathrm{Var}[U] &= \overline{U^2} - (E[U])^2, \text{where} \\ \overline{U^2} &= (m+1)(2m+1)t^2/6, \text{and} \end{aligned} \tag{15}$$

$$\begin{aligned} E[V] &= T \sum_{j=0}^{k-1} q(j) \sum_{i=0}^{j} \frac{i+1}{j+1}, \\ \mathrm{Var}[V] &= \overline{V^2} - (E[V])^2, \text{where} \\ \overline{V^2} &= T^2 \sum_{j=0}^{k-1} q(j) \sum_{i=0}^{j} \frac{(i+1)^2}{j+1}, \end{aligned}$$

and $q(j)$ is given in (11).

The mean and variance of packet delay are then calculated by substituting (13), (14), (15) into (12).

 ISBN: 978-1-61208-633-0

IV. NUMERICAL RESULTS

In this section, we validate the analytical model by simulation. For this purpose, we have built a discrete event simulator by C++ to generate simulation results. The duration of each simulation run is 5,000 seconds, with a warm-up period of 500 seconds. From Figures 3, 4(a) and 4(b), it can be seen that simulation and analytical results match quite well. It confirms that our model is sufficiently accurate. At the same time, we also use the developed analytical model to investigate the impact of different parameters on the performance of the MAC protocol.

The system parameters used throughout this section are as follows. With a 25 MHz spectrum, the 64-QAM modulation scheme is used to achieve a data rate of 120 Mbps. A *mini slot* is a basic unit of different time slots, and has a duration of $\frac{1}{2500}$ millisecond. Each bandwidth request has a duration of 6 mini slots. Each data slot has a duration of 94 mini slots, which allows the transmission of approximately 0.5 KB data at 120 Mbps. The information subframe has a duration of 10 request slots. Also, $W = 8$, $r = 3$, and $R = 5$.

First, we set $m = d = 12$, and evaluate the failure probability of a request for different N. The result is shown in Figure 3. As expected, the more UE in the network, the higher the collision probability that requests would experience.

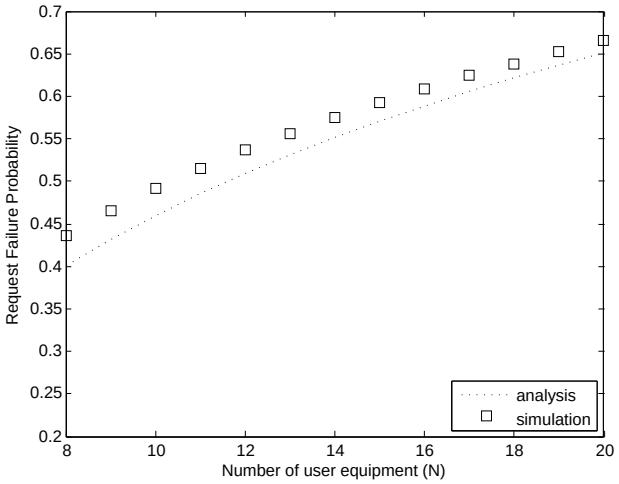
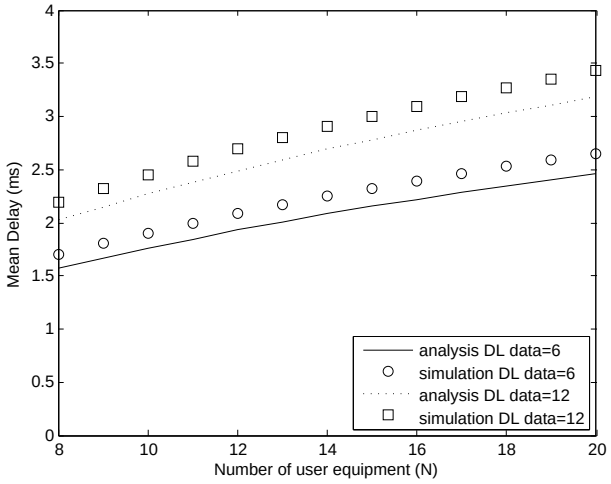


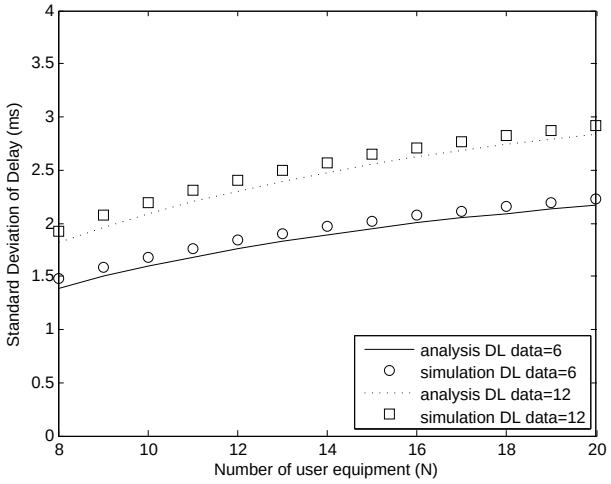
Figure 3. Failure probability of request ($b = 6, m = d = 12$).

Having the probability of unsuccessful request, the mean and standard deviation of the packet delay are computed as described in Section III-B. These analytical values together with the simulation results are plotted against N in Figures 4(a) and 4(b). As shown in the figures, reducing the number of data slots available in the full duplex data subframe causes an increasing in both mean and standard deviation of the packet delay. This is because when there are less data slots available, the failure probability of requests increases, and thus the average backoff time increases. As a result, the mean and standard deviation of packet delay increase.

Next, we investigate the impact of the size of the downlink data subframe. We consider that the traffic is close to symmetric and, only occasionally, the BS has to rely on the downlink data subframe to deliver packets to UE. In that case,



(a)



(b)

Figure 4. (a) Mean delay and (b) standard deviation of delay versus the number of user equipment ($b = 6, m = 12$).

the number of data slots in this subframe can be much smaller than d. Here, we use the case of $b = 12, m = 12, N = 20$ as a reference, and calculate how much (measured in %) delay is reduced when b is reduced from 12 to 2. The results are plotted in Figure 5.

It can seen that when $b = 2$, the mean delay is reduced by about 37%. Clearly, the reduction of delay is due to a smaller frame size as t_h decreases with b. This reflects one of the benefits of the full duplex MAC protocol.

V. CONCLUSION

In this paper, we have proposed a MAC protocol for mobile telecommunication networks which support full duplex wireless communication. In this MAC protocol, binary exponential backoff is used to resolve contention. We have also developed an analytical model for performance evaluation in terms of request failure probability, mean and standard deviation of packet delay. Explicit forms of these performance metrics have

 ISBN: 978-1-61208-633-0

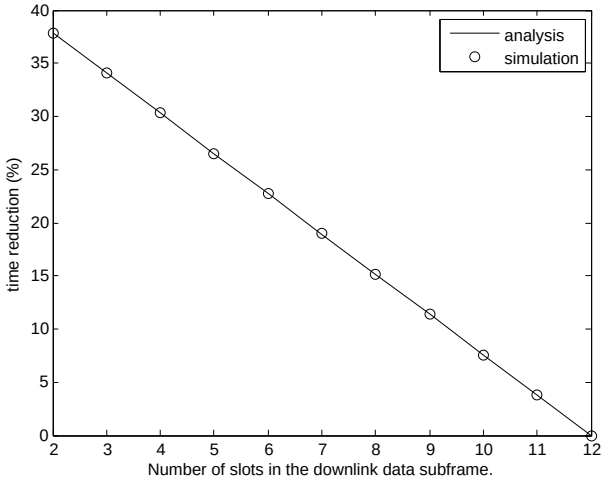


Figure 5. Percentage of delay reduction versus the number of slots in downlink data subframe.

been derived. The model has been validated by simulation results. The numerical results have shown the impact of various parameters on the performance metrics. Moreover, packet delay can be significantly reduced due to the deployment of full duplex wireless communication.

REFERENCES

[1] E. B̈jornson, E. G. Larsson, and T. L. Marzetta, "Massive MIMO: Ten Myths and One Critical Question," IEEE Communications Magazine, vol. 54, no. 2, February 2016, pp. 114-123.

[2] L. Dai *et al.*, "Non-orthogonal Multiple Access for 5g: Solutions, Challenges, Opportunities, and Future Research Trends," IEEE Communications Magazine, vol. 53, no. 9, September 2015, pp. 74-81.

[3] C. Xu, Y. Hu, C. Liang, J. Ma, and Li Ping, "Massive MIMO, Non-orthogonal Multiple Access and Interleave Division Multiple Access," IEEE Access, vol. 5, August 2017, pp. 14728–14748.

[4] M. Jain *et al.*, "Practical, Real-time, Full Duplex Wireless," In Proceedings of ACM MobiCom'11, Sep 19-23, 2011, Las Vegas, pp. 300–312.

[5] M. Duarte *et al.*, "Design and Characterization of a Full-duplex Multiantenna System for Wifi Networks," IEEE Trans. on Vehicular Technology, vol. 63, no. 3, March 2014, pp. 1160-1177.

[6] K. Miura and M. Bandai,"Node Architecture and Mac Protocol for Full Duplex Wireless and Directional Antennas," In Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'12), Sep 9-12, 2012, Sydney, pp. 385-390.

[7] W. Cheng, X. Zhang, and H. Zhang, "RTS/FCTS Mechanism Based Full-duplex Mac Protocol for Wireless Networks," In Proceedings of IEEE Globecom 2013, Dec 9-13, 2013, Alanta, pp. 5017–5022.

[8] Q. Qu *et al.*, "Fuplex: A Full Duplex Mac for the Next Generation Wlan," In Proceedings of the 11th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE 2015), Aug 19-20, 2015, Taipei, pp. 239-245.

[9] W. Choi, H. Lim, and A. Sabharwal, "Power-controlled Medium Access Control Protocol for Full-duplex Wifi Networks," IEEE Trans. on Wireless Communications, vol. 14, no. 7, July 2015, pp. 3601-3613.

[10] A. Tang and X. Wang, "A-duplex: Medium Access Control for Efficient Coexistence Between Full-duplex and Half-duplex Communications," IEEE Trans. on Wireless Communications, vol. 14, no. 10, Oct 2015, pp. 5871-5885.

[11] W. Zhou, K. Srinivasan, and P. Sinha, "RCTC: Rapid Concurrent Transmission Coordination in Full Duplex Wireless Networks," In Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP), Oct 7-10, 2013, Göttengen, pp. 1-10.

[12] D. Marlal and Ö Gürbüz, "S-CW FD: A Mac Protocol for Full-duplex in Wireless Local Area Networks," In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2016), Mar 3-6, 2016, San Francisco, pp. 1–6.

[13] Y. Liao, K. Bian, L. Song, and Z. Han, "Full-Duplex MAC Protocol Design and Analysis," IEEE Communications Letters, vol. 19, no. 7, July 2015, pp. 1185–1188.

[14] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, March, 2000, pp. 535–547.

 ISBN: 978-1-61208-633-0

DDoS Attacks as an Important Network Phenomenon

Ilija Basicevic
and Miroslav Popovic

Faculty of Technical Sciences
University of Novi Sad, Serbia
Email: ilibas@uns.ac.rs
Telephone: (+381) 21 4801 242

Stanislav Ocovaj

RT-RK Institute for Computer Based Systems
Novi Sad, Serbia
Email: stanislav.ocovaj@rt-rk.com

***Abstract*—This paper presents several aspects of Distributed Denial of Service (DDoS) attacks which have been an important network phenomenon since the end of 1990s. A short introduction to this topic is given here. DoS mechanisms, such as reflection and amplification, are explained including example attacks. Also, DNS fast flux as a technique for hiding command and control communication (C&C) is described. A short overview of entropy based DDoS detection is given, and an example method explained. Certain methods for attack mitigation (most notably puzzles) are explained, as well.**

Keywords–network security; Distributed denial of service attacks (DDoS); entropy; DNS amplification attack; SYN Flood attack.

I. INTRODUCTION

DDoS attacks have already become part of the Internet landscape. Although unwanted part, there are currently no signs that they will go away. As it is well known, the aim of a DDoS is to stop or interrupt or slow down the operation of an Internet server (most often a web server, but it can be an e-mail, DNS, or other type of server). There are many ways to inhibit the operation of a computer or a network service, but Internet DoS attacks typically achieve that by depleting resources.

DoS attacks are important because of the impact they have on operation of companies that rely on Internet in their communication with customers. The targets of DoS are not only business companies but government and nongovernment agencies, too.

The first distributed DoS (DDoS) attack was registered in 1999. The attacker used Trinoo tool to attack University of Minnesota computer network [1]. Since then, the damages they inflict to business operations have been constantly rising.

The size of DoS attacks that is seen on the Internet keeps rising every year. In 2012, attacks of the size of 70 Gbps of noisy traffic were seen. In 2013, the largest DoS attack was 300 Gbps. Early in 2014, the attack of 400 Gbps was reported. In 2016, the attack launched by Mirai botnet surpassed 1 Tbps.

Criminal groups that extort money from business companies are often behind DDoS attacks. Companies have reported losing up to $100000 (some even more) per downtime hour during DoS attacks [2]. Companies often decide to pay criminals because the law procedure is slow and criminal groups are often in different countries, which complicates the procedure. There are also politically motivated attacks, which are often led by cyber-terrorists or by countries in conflict. For an example of politically motivated attack, see [3] about DoS attack in Estonia in 2006.

Another purpose of DoS is that it can be used (and is used more and more) to cover up traces of an intrusion - as a distraction mechanism or, to explain in a more figurative way, as a smoke shield. While security and IT officers of a company deal with DoS, the attackers can more easily intrude the company information system undetected.

Having in mind the protocol stack reference model (ISO OSI), the attack can be realized at different levels of the protocol stack.

II. DISTRIBUTED DOS

As already noted, there are many ways to produce denial of service effect. Distributed DoS attacks are usually realized either as:

- Bandwidth attacks attack on the availability of network links to the server, or as
- Resource starvation attacks attack on the availability of resources (e.g. memory) at the server.

Some well-known examples of resource starvation attacks are:

- Sending XML documents with deeply nested schemes, and
- Sending large number of packets that require crypto operations, as those are usually resource intensive.

DoS attacks at network servers are usually realized in a distributed manner (Distributed DoS, DDoS). Typically, there is a large number of computers (thousands or tens of thousands) that are connected to the Internet, and that take part in the attack. The owners of computers are usually not aware that their computers participate in the attack. The computers (sometimes called zombies) are usually infected by malware which installs DoS agents on them – thus they get under the control of the attacker. These computers form a bot net, which is controlled by the attacker.

The aim of attackers is to avoid tracing the IP address of the command computer (and its owner) and special measures are taken to hide the so called Command&Control communication between bot net computers and the attacker.

A. Mathematical model of DDoS

In [4], authors model the system under SYN flood DDoS attack as a two-dimensional queuing model with N servers, two arrival processes and two service times of different distribution. Both the arrival of regular request packets and the arrival of attack packets are modeled as Poisson processes, but with

 ISBN: 978-1-61208-633-0

different arrival rates λ_1 and λ_2. This is in accordance with the prevalent view on properties of Internet traffic [5], [6]. At most N half-open connections are allowed at one moment. Half-open connection for a regular request packet is held for random time which is exponentially distributed. The two arrival processes are independent of each other and of holding times for half-open connections. Based on these assumptions, DDoS is modeled as two-dimensional embedded Markov chain. The model allows calculation of security metrics such as:

- Connection loss probability, and
- Buffer occupancy percentage of half-open connections for regular traffic.

B. DNS fast flux

Domain name system (DNS) fast flux is a technique that attackers can use to hide the C&C center. There are two variants of this technique: single and double fast flux. In the single fast flux, a large set of addresses (hundreds or thousands) is associated with a domain name. These addresses are swapped in and out (for example in a round robin fashion) with high frequency each set of assigned addresses is changed after less than 5 minutes. The computer that connects to a web server every 5 minutes will connect each time to a different address.

In the double fast flux, the same is done as in the single version, but this time with an authoritative name server responsible for the entire DNS zone (containing multiple domains) and not only with a single domain name, as in the single fast flux. This time, the IP address of authoritative Name Server is also changing constantly. This gives an additional layer of protection to the attacker.

C. Reflection and amplification

Reflection is an important mechanism that is used by certain DoS attacks. In that case, the attack traffic is sent to a reflector that replies by sending messages to the source address of received messages, which is falsified (spoofed), and points incorrectly to the target of the attack. Thus, the reflected traffic is directed to the target. Amplification is the effect that the attacker tries to achieve: traffic to the target should be larger than the traffic sent by attackers by some amplification factor. Well-known reflected attacks are:

- Smurf (realized by sending Internet Control Message Protocol (ICMP) Echo Request -ping- with spoofed source address to the broadcast address), Figure 1 and
- Fraggle (realized by sending User Datagram Protocol (UDP) echo with spoofed source address to the broadcast address).

These two attacks presented a real threat in 1990s but today, most of the networks are configured so as not to be vulnerable to those two attacks (IP-directed broadcast is disabled at routers).

D. DNS amplification attack

In this type of attack, the attacker uses publicly accessible DNS servers to flood the target with DNS response traffic. Members of the botnet send a large number of DNS name lookup requests to the DNS server with spoofed source address. Usually, DNS requests are for many names and with the type ANY so that responses are larger and the amplification factor is increased. The ANY type requires all known information to be returned for the specified name. As the response traffic originates from public DNS servers and as it is legitimate data, it is hard to prevent it.

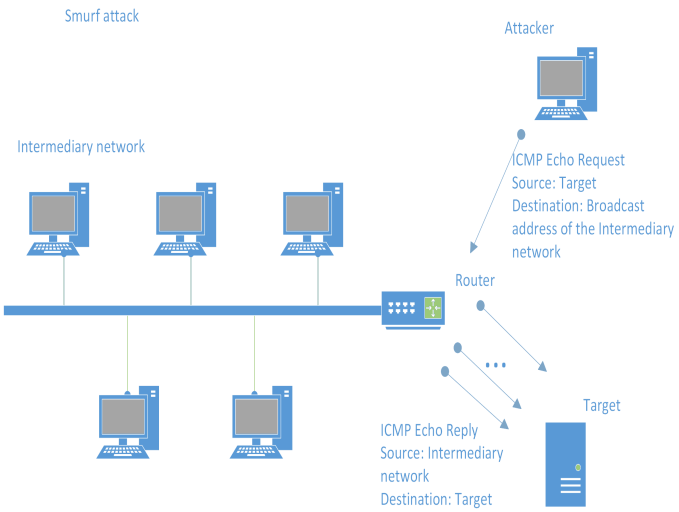


Figure 1. Smurf DDoS attack.

Variations of this attack can include attackers compromising the DNS server and adding large TXT RR records – to increase the amplification factor. TXT RR allows arbitrary text to be inserted in the DNS record. There are existing legitimate large records, so this step is not necessary.

E. SYN Flood attack

This is one of the most frequent DoS attacks for several years and probably the most investigated attack in scientific publications. Boteanu and Fernandez call it Mother of all DoS attacks [7]. In the course of the attack, attackers initiate a large number of Transmission Control Protocol (TCP) three-way handshakes but do not actually complete them.

Thus, resources at the server that are allocated for connection establishment and in normal case released when the connection enters the active state (or is prematurely closed), stay allocated, which leads to resource starvation. Upon receiving the SYN packet, the TCP server side enters the SYNRECEIVED state and starts the timer with the duration of typically 75 s [8].

The attack does not aim to overload the end hosts memory but simply to exhaust the so-called backlog of half-open connections associated with a port number. Backlog is a system limit on a number of TCP Control Block (TCB) structures that can be resident at any time [9]. To achieve the desired effect, the attacker should send new barrage of connection requests as soon as the attacked system starts to reclaim TCB blocks allocated during the previous barrage. The frequency of attack has to be adapted to the TCB reclamation timer. The greater frequency increases the risk of detection without adding to the effectiveness of the attack.

 ISBN: 978-1-61208-633-0

F. Shrew attack

This is a low rate attack that uses attack stream of a square waveform [10] with the following parameters: period T, burst length L, peak rate R. It has the following properties [11]:

- R is enough to exceed link capacity in combination with baseline traffic
- L is long enough to induce timeout (typically greater than round trip time RTT)
- T is scaled in accordance with the minimum retransmission timeout (RTO)

The logic behind the attack is to let TCP module detect that the link is congested. After the initial attack burst, the TCP will wait until the expiry of retransmission timeout. When it does retransmit, it will collide with one of the subsequent attack bursts. As a result, the TCP can experience very low (near zero) throughput and connection close.

III. Methods for detection of DDoS

There are two main classes of DoS attack detection methods: volume based and feature based.

- Feature-based methods rely on inspection of packet headers.
- Volume-based methods monitor changes in traffic volume.

The attackers goal is to achieve DoS effect with as little attack traffic as possible – in order to avoid detection. Having that in mind, the importance of feature-based detection is easier to understand. There are known advantages of feature-based over volume-based methods in detection of small volume attack traffic [12].

A. Entropy based detection

Among feature based methods, entropy has a very important place. It is obvious that network events, such as DDoS attacks and port scans, change the randomness of packets addresses and ports. The most important advantage of entropy is its generality. An entropy based method can detect a wide range of network traffic anomalies (including DDoS attacks, but also flash crowds, and other types of events). Ref [13] presents a comparison of the entropy based method for detection of DDoS attacks and a custom tailored method for detection of SYN Flood attacks [14]. The result of the comparison is that with respect to detection performance, the entropy based method comes close to custom tailored one in detection of SYN Flood attacks, but on the other hand the entropy based method can detect UDP DDoS as well, while the custom tailored method is completely unusable in detection of attacks other than SYN Flood.

Thus, the number of research papers that propose the use of entropy is not small, and in the continuation, we will mention some of them.

Authors experimented with the use of Shannon entropy ([15], [16], [13] and others):

$$H(Z) = -\sum_{i=1}^{N} p(z_i) log(p(z_i)), \quad (1)$$

where $p(z_i)$ is probability that Z takes the value z_i. The entropy value is then normalized by the log (N), where N is the number of distinct z_i values that appear in the observed interval. All logarithms are with the base 2, because the information is represented in bits.

Shannon entropy has following properties:

- Nonnegativity

$$H(Z) \geq 0, \forall p(z_i) \in [0, 1], \quad (2)$$

- Symmetry

$$H(p(z_1), p(z_2), ...) = H(p(z_2), p(z_1), ...), \quad (3)$$

- Maximality

$$H(p(z_1), p(z_2), ..., p(z_n)) \leq H(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}), \quad (4)$$

- Additivity

$$H(x, y) = H(x) + H(y), \quad (5)$$

if x and y independent variables.

Another entropy formula that is attractive to researchers is Tsallis entropy [17]:

$$H(Z) = \frac{1 - \sum_{i=1}^{N} p_{z_i}^q}{q - 1}, \quad (6)$$

the value of H is in the range (0, H_q^{max}), and

$$H_q^{max} = \frac{1 - N^{1-q}}{q - 1}, \quad (7)$$

Shannon entropy is extensive, while Tsallis is not – it does not have the property of additivity. For q tending to 1, Tsallis formula converges to Shannon. Tsallis has proposed the formula for the investigation of the properties of systems that exhibit fractal and long range dependent behavior. Accordingly, the motivation for the use of Tsallis formula in detection of network traffic anomalies is the well-established notion that Internet traffic is self-similar and long range dependent. Those properties have been discovered more than twenty years ago [18], [19], [20]. Earlier view is that self-similarity is stronger with the increase of network utilization [18]. More recent view is that with a traffic increase, inter arrival process tends to Poisson and packet sizes to be independent [5]. Also, it is noted that on sub second time scales, network traffic can be well represented by the Poisson model [6].

One of the early papers on the use of Tsallis entropy in detection of DDoS is [21]. Tellenbach et al [22] have proposed the use of Traffic Entropy Spectrum (TES) that is based on the use of Tsallis entropy. In ref [23], performance of Tsallis formula is compared to the one of Shannon formula. The conclusion is that Tsallis formula can outperform Shannon in detection of DDoS attacks, but that requires careful tuning of Tsallis Q parameter and there is no universal Q value that performs best in all detection scenarios.

Another generalization of Shannon entropy is Renyi entropy. Renyi entropy is extensive, and for α tending to 1, it converges to Shannon entropy.

$$H(Z) = \frac{1}{1 - \alpha} \log \sum_{i=1}^{N} p(z_i)^{\alpha}, \quad (8)$$

In refs [24], [25] and [26] authors have compared the performance of those formulas. In [24], authors conclude that

with respect to DoS detection, generalized entropy measures outperform Shannon. In [25], authors conclude that generalized entropies and feature-based distributions perform better than Shannon entropy and counter-based methods. The important conclusion of their research is that for successful detection of different anomalies, a wide range of distributions should be used.

To determine the extent of changes between observed and assumed distributions, Kullback-Leibler divergence (K-L) can be used [26]. The formula for K-L divergence is:

$$D_{KL}(p,q) = \sum_{i=1}^{N} p(i) log(\frac{p(i)}{q(i)}, \quad (9)$$

It is also used with the maximum entropy, the application of which is proposed by some researchers, see [27], [28].

The change of randomness is not enough to certainly indicate an attack. Some limitations of entropy are overcome with the use of Kolmogorov complexity [29]. There is a known problem with efficient calculation of Kolmogorov complexity, which today prevents its use in online detection. As an alternative, researchers propose the use of Titchener (T-entropy) [30].

The entropy is calculated on certain distributions of network traffic parameters. Those distributions can be simple (e.g. source and destination addresses/ports) or complex. The most important complex distribution is a flow size distribution (FSD). Flow is a sequence of packets exchanged between two endpoints. The endpoints are defined with a 5-tuple (SrcIp, SrcPort, DstIp, DstPort, Protocol) containing source and destination addresses and ports, and the protocol used for communication. In case of TCP, the flow corresponds to TCP connection. In case of UDP, the flow is defined using maximum allowed time between two consecutive packets in a flow. A comparison on the usability of FSD versus distribution of addresses is given in [31], where it is concluded that FSD in certain scenarios outperforms simple packet distributions.

B. Change point detection

The detection method used in [13], [31], [23] assumes that entropy time series are subject to change point detection algorithm, more specifically CUSUM [32]. On the other hand, the approach in [27] avoids the use of change point detection.

CUSUM is used in many application fields and it is based on hypothesis testing. The input time series are independent and identically distributed random variables that are bounded by a finite value. Two hypotheses define distribution before and after the change. The formulas used in [13], [31], [23] are given here:

$$\mu_n = \beta_1 y_n + (1-\beta_1)\mu_{n-1} \quad (10)$$

$$d_n = max\left\{0, d_{n-1} + y_n - (\mu_n + K)\right\}, d_0 = 0 \quad (11)$$

$$\sigma_n^2 = \beta_2 (y_n - \mu_n)^2 + (1-\beta_2)\sigma_{n-1}^2, H = h\sigma_n \quad (12)$$

If $d_n > H$, a change is reported ($g_n = 1$). Thus, H is the decision threshold. μ_n is an estimation of average value of time series y_n. Counter d_n accumulates deviations of y_n from μ_n that are greater than K. K is typically set to the half of the minimum shift to be detected, here it is set to 0.3. β_1 and β_2 are EWMA adaptation factors - their values are 0.2 and 0.05 respectively. σ_n is an estimation of the standard deviation of time series y_n.

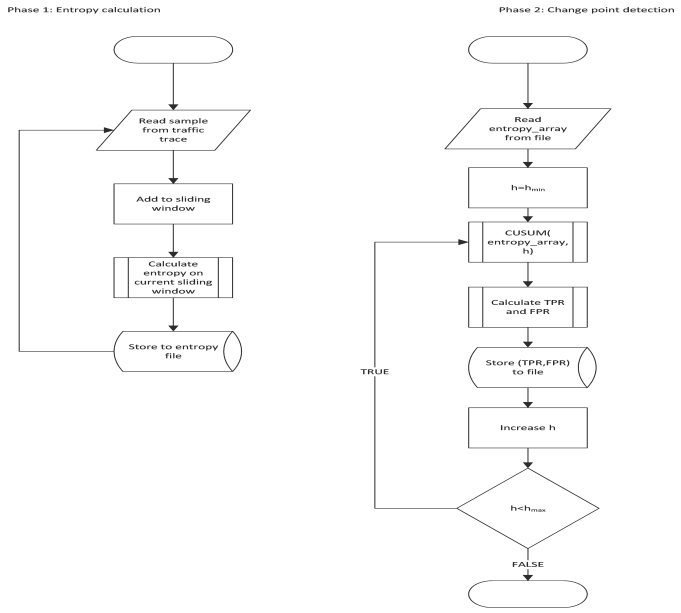


Figure 2. An example of the detection method.

C. The overall method of detection

As an example, the method of detection used in [13], [31], [23] is presented in Figure 2. The traffic trace file is input to the entropy calculation program. For the calculation of entropy, sliding window of 1 s (10 sub-intervals of 0.1 seconds) is used. The output of entropy calculation program is a file containing entropy time series. In the next step, CUSUM is applied to entropy time series for change point detection. The CUSUM h parameter is iterated in the range [0, 20]. For each value, detection is performed and true positive rate (TPR) and false positive rate (FPR) are calculated and stored in a file. The file is used in the last phase for the creation of receiver operating curves (RoC).

The detector perfoemance can be further improved with the use of Takagi-Sugeno-Kang fuzzy method, see [33].

D. The dataset problem

Basically, there are two options: simulations and analysis of real traffic traces. The problem with the second one, which is somehow more appealing for a scientist, is that the number of existing traces available online is rather small and, to the best of our knowledge, there is no existing labeled real network trace that contains both baseline and DoS attack traffic. So, researchers most often opt for two approaches:

- Simulations (ns2 [34], ns3 [35], omnet++ [36], etc.) and
- Injection of artificial DoS traffic to real network traces.

The problem with simulations is, as always, how realistic they are. The problem with injection, is (among other things), that in such a trace is not visible the reaction of the target server and other network subjects to the DoS i.e. the server is less responsive to legitimate clients during the attack thus, they repeat connection requests, etc.

Some of real datasets that are still used in research of DDoS are outdated (e.g. [37]). For an overview of existing datasets, see [38]. Another possibility are emulation testbeds, such as

 ISBN: 978-1-61208-633-0

[39] and [40]. They integrate simulation and real systems, using soft routers.

IV. The Reaction to DDoS - Attack Mitigation

The defense mechanisms can be classified into response mechanisms (primarily filtering, rate-limiting, and capability methods) and tolerance mechanisms (congestion policing, fault tolerance, and resource accounting). Tolerance mechanisms do not rely on attack detection, but in some cases, they are expensive and lead to inefficient use of overprovisioned resources (the case of fault tolerance methods). For a more comprehensive overview of defense mechanisms, see [41].

The first defense measure that has been applied by Internet Service Providers (ISPs) and carriers (for years) has been blackholing the target IP address. The router that has detected a DoS attack will blackhole the target IP address. All traffic destined to that address (normal and attack) will be discarded. The goal of the attack is achieved completely but on the other hand, other hosts in the network have been spared.

A more recent defense technique is based on the use of a scrubbing center. The router that detects DoS attack reroutes the traffic through the scrubbing center. The scrubbing center is typically located in a cloud and it will remove attack packets from the traffic. The problem with out-of-band scrubbing centers is that re-routing decision is done by a human analyst. Thus, there is time required for attack mitigation to take place. A series of short attacks can evade the detector.

One of resource accounting mechanisms attempts to balance the workload between clients and servers by introducing cryptographic puzzles, Figure 3. In that case, the client is required to solve the puzzle before the server allocates resources to processing the clients request.

Figure 3 shows the order of messages in a system which supports puzzles. Upon receiving the request from the client, the protected server forwards the request to the puzzle generator. The puzzle generator generates a puzzle and sends it to the client. When it solves the puzzle, the client sends the solution to the puzzle generator, which checks the solution. Only when the puzzle has been solved, the puzzle generator informs the protected server, which continues the communication with the client.

The puzzles use cryptographic mechanisms that make it very hard for clients to avoid solving the puzzle. Even in such a case, attackers that control large botnets are in a much better position than legitimate clients and actually, the targeted balance of workload between the server and the attacker is not achieved.

A more promising approach is the use of guided tour puzzles. The limiting factor in this case is not the clients Central Processing Unit (CPU) power but the round trip time of the network path, which the attacker cannot overcome by multiplying CPU resources [42].

V. Conclusion

The paper presents the perspective of distributed denial of service attacks. A short introduction is given, including the motivation, size and historical beginnings of this network phenomenon. Important mechanisms that are used by attackers (reflection, amplification, hiding of command and control communication) are described and examples are given. A short overview of entropy based DDoS attack detection is provided as well as a description of certain attack mitigation techniques (including scrubbing centers and puzzles).

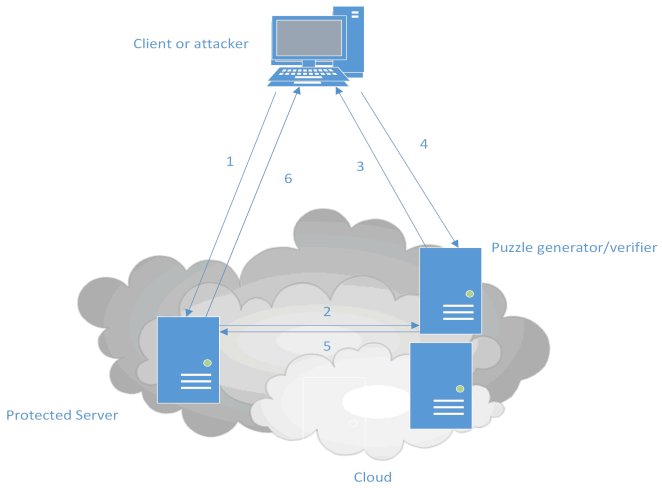


Figure 3. Message exchange in a system which supports puzzles.

Acknowledgment

This work has been partially supported by the Ministry of Education and Science of the Republic of Serbia under the Project TR32031.

References

[1] "DDoS attacks history," URL: https://security.radware.com/ddos-knowledge-center/ddos-chronicles/ddos-attacks-history/ [retrieved: 1, 2018].

[2] T. Matthews, "Incapsula Survey : What DDoS Attacks Really Cost Businesses," 2014.

[3] "A brief history of DDoS attacks," 2016, URL: https://eugene.kaspersky.com/2016/12/06/a-brief-history-of-ddos-attacks/ [retrieved: 1, 2018].

[4] Y. Wang, C. Lin, Q.-L. Li, and Y. Fang, "A queueing analysis for the denial of service (dos) attacks in computer networks," Comput. Netw., vol. 51, no. 12, Aug. 2007, pp. 3564–3573.

[5] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "Internet traffic tends to poisson and independent as the load increases," Bell Labs, Tech. Rep., 2001.

[6] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary poisson view of internet traffic," in Proceedings of IEEE INFOCOM, 2004, pp. 1558–1569.

[7] D. Boteanu and J. M. Fernandez, "A comprehensive study of queue management as a DoS counter-measure," International Journal of Information Security, vol. 12, no. 5, 2013, pp. 347–382.

[8] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 2002, pp. 1530–1539.

[9] W. Eddy, "TCP SYN flooding attacks and common mitigations," 2007, rfc: RFC 4987.

[10] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03), 2003, pp. 75–86.

[11] C.-W. Chang, S. Lee, B. Lin, and J. Wang, "The taming of the shrew: Mitigating low-rate tcp-targeted attack," IEEE Transactions On Network Service Management, vol. 7, no. 1, 2010.

 ISBN: 978-1-61208-633-0

[12] P. Du and S. Abe, “Detecting dos attacks using packet size distribution,” in 2007 2nd Bio-Inspired Models of Network, Information and Computing Systems, 2007, pp. 93–96.

[13] I. Basicevic, S. Ocovaj, and M. Popovic, “Evaluation of entropy-based detection of outbound denial-of-service attacks in edge networks,” Security and Communication Networks, vol. 8, 2015, pp. 837–844.

[14] V. A. Siris and F. Papagalou, “Application of anomaly detection algorithms for detecting syn flooding attacks,” in Globecom, 2004, pp. 2050–2054.

[15] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, “An empirical evaluation of entropy-based traffic anomaly detection,” in Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, 2008, pp. 151–156.

[16] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” in Proceedings of the ACM SIGCOMM 2005, 2005, pp. 217–228.

[17] C. Tsallis, “Possible generalization of boltzmann-gibbs statistics,” Journal of Statistical Physics, vol. 52, no. 1-2, 1988, pp. 479–487.

[18] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, “On the self-similar nature of ethernet traffic,” in SIGCOMM ’93 Conference proceedings on Communications architectures, protocols and applications, 1993, pp. 183–193.

[19] M. Crovella and A. Bestavros, “Self-similarity in world wide web traffic: evidence and possible causes,” in Proceedings of the 1996 ACM SIGMETRICS international conference on measurement and modeling of computer systems, 1996, pp. 160–169.

[20] V. Paxson and S. Floyd, “Wide area traffic: the failure of poisson modeling,” IEEE/ACM Transactions on Networking, vol. 3, no. 3, 1995.

[21] A. Ziviani, A. T. A. Gomes, M. L. Monsores, and P. S. S. Rodrigues, “Network anomaly detection using nonextensive entropy,” IEEE Communications Letters, vol. 11, no. 12, 2007.

[22] B. Tellenbach, M. Burkhart, D. Sornette, and T. Maillart, “Beyond shannon: Characterizing internet traffic with generalized entropy metrics,” in Proceedings of the 10th International Conference on Passive and Active Network Measurement, 2009, pp. 239–248.

[23] I. Basicevic, S. Ocovaj, and M. Popovic, “Use of tsallis entropy in detection of syn flood dos attacks,” Security and Communication Networks, vol. 8, 2015, pp. 3634–3640.

[24] C. Lima, F. de Assis, and C. de Souza, “A comparative study of use of shannon, rnyi and tsallis entropy for attribute selecting in network intrusion detection,” in Intelligent Data Engineering and Automated Learning - IDEAL 2012. Lecture Notes in Computer Science, vol 7435. Springer, Berlin, Heidelberg, 2012.

[25] P. Bereziski, J. Pawelec, M. Maowidzki, and R. Piotrowsk, “Entropy-based internet traffic anomaly detection: A case study,” in Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX, 2014, pp. 47–58.

[26] P. Berezinski, B. Jasiul, and M. Szpyrka, “An entropy-based network anomaly detection method,” Entropy, no. 17, 2015, pp. 2367–2408.

[27] Y. Gu, A. McCallum, and D. Towsley, “Detecting anomalies in network traffic using maximum entropy estimation,” in 5th ACM SIGCOMM conference on Internet measurement (IMC ’05), 2005, pp. 32–32.

[28] A. Coluccia, “Glrt-based change detection for non-stationary data via maximum entropy,” in GTTI Annual Meeting, 2010.

[29] A. Wagner and B. Plattner, “Entropy based worm and anomaly detection in fast ip networks,” in 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05), 2005, pp. 172–177.

[30] U. Speidel, R. Eimann, and N. Brownlee, “Detecting network events via t-entropy,” in 6th International Conference on Information, Communications Signal Processing, 2007, pp. 1–5.

[31] I. Basicevic, S. Ocovaj, and M. Popovic, “The value of flow size distribution in entropy-based detection of dos attacks,” Security and Communication Networks, vol. 9, 2016, pp. 958–965.

[32] E. S. Page, “Continuous inspection scheme,” Biometrika, vol. 41, no. 1/2, 1954, pp. 100–115.

[33] M. Petkovic, I. Basicevic, D. Kukolj, and M. Popovic, “Evaluation of takagi-sugeno-kang fuzzy method in entropy-based detection of ddos attacks,” Computer Science and Information Systems, vol. 15, no. 1, 2016, pp. 139–162.

[34] “The network simulator - ns2,” http://www.isi.edu/nsnam/ns [retrieved: 3, 2018].

[35] “The network simulator - ns3,” https://www.nsnam.org/ [retrieved: 3, 2018].

[36] “Omnet++ discrete event simulator,” https://www.omnetpp.org/ [retrieved: 3, 2018].

[37] “M. i. t. lincoln laboratory, darpa intrusion detection evaluation data set,” https://www.ll.mit.edu/ideval/data/ [retrieved: 3, 2018], 2018.

[38] S. Behal and K. Kumar, “Trends in validation of ddos research,” Procedia Computer Science, vol. 85, no. 2016, 2016, pp. 7–15.

[39] “Emulab - network emulation testbed home,” https://www.emulab.net/ [retrieved: 3, 2018].

[40] “The deter project,” https://deter-project.org/ [retrieved: 3, 2018].

[41] M. Abliz, “Internet denial of service attacks and defense mechanisms,” 2011, report: TR-11-178, University of Pittsburgh.

[42] M. Abliz, T. Znati, and A. Lee, “Mitigating distributed service flooding attacks with guided tour puzzles,” International Journal on Advances in Security, vol. 5, 2012, pp. 121–133.

 ISBN: 978-1-61208-633-0

Social-aware Peer Discovery for Small Cell based Device-to-Device Communications

Aamir Nadeem and Ho-Shin Cho
School of Electronics Engineering
Kyungpook National University
Daegu, South Korea
e-mail: aamirnadeem04@gmail.com and hscho@knu.ac.kr

***Abstract*—In this paper, we propose a peer discovery method for Device-to-Device (D2D) communications using Multi-Attribute Decision Modeling (MADM) technique. The method exploits both physical and social characteristics of User Equipment (UE) to find the most suitable partner for D2D communication. A small-cell based scenario is considered with the aim to pair the UEs with similar social and physical characteristics. Various social and physical attributes are used to rank the UEs and select the best one as a D2D partner. Simulation and analysis results are provided for various performance parameters.**

Keywords-Device-to-Device; User Equipment; Peer Discovery; TOPSIS; Small Cell etc.

I. INTRODUCTION

Small-Cell Network (SCN) and Device-to-Device (D2D) communications are believed to be the emerging solutions for the heavily loaded cellular networks [1]. Since both paradigms aim to provide low cost, low power and short-range communication and reduce the burden on Macro-Cell Base-Station (MCBS), they can be integrated to exploit both the licensed and unlicensed spectrum bands for direct communication between Cellular User Equipment (CUEs).

In order to establish a D2D link between two CUEs, the source UE needs to discover the target UE with the desired service. The peer discovery may be performed with or without the assistance of Base-Station (BS) or eNodeB (Evolved Node B). The former method can improve the peer discovery process at the cost of higher signaling overhead. Similarly, the latter approach can achieve scalability and out-of-coverage discovery but lacks synchronization and collision avoidance [2].

In this work, we propose a method based on Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [3] for social-aware D2D peer discovery in SCN environment. It combines both the social and physical attributes of a UE to determine a suitable partner for D2D communication. A decision matrix is constructed where the UEs connected to the Small-Cell Base Station (SCBS) are considered as alternatives and various social and physical attributes as constraints. The decision matrix is then normalized and used to calculate the positive and negative ideal solutions. Then, based on the distance from the two ideal solutions, the candidate UEs are ranked and the best one is selected for D2D communication. The SCN based scenario can overcome the issues in traditional macro-cell based D2D communication. Also, the method is easy to implement and power-efficient as the peer discovery process is performed by the SCBS rather than the UE itself.

The next section presents the problem definition and formulation. The third section discusses the simulation model and results and the final section summarizes the discussion with conclusion and future direction.

II. PROBLEM DEFINITION AND FORMULATION

D2D peer discovery is based on the physical and social relationship between human users. Users who are friends on social networks with same age group and profession are more likely to have similar choices. Moreover, for D2D communication the users also need to be located closely. Therefore, we consider both social and physical attributes to maximize the chances of willingness for D2D communication. For this purpose, we use TOPSIS method which ranks the available alternatives, i.e., UEs and select the best option based on the given attributes. The basic steps of the method are as follows.

Step 1: We construct a decision matrix D_m in which the rows consist of UEs as alternatives while the columns are based on various social and physical attributes of the corresponding UE as given in (1).

$$D_m = \begin{array}{c} \\ UE_1 \\ \vdots \\ \vdots \\ \vdots \\ UE_m \end{array} \begin{array}{c} \begin{array}{cccccc} At_1 & At_2 & At_3 & \dots & At_{p-1} & At_p \end{array} \\ \begin{bmatrix} X_{11} & X_{12} & X_{13} & \cdots & X_{1p-1} & X_{1p} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2p-1} & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{m1} & X_{m2} & X_{m3} & \cdots & X_{mp-1} & X_{mp} \end{bmatrix} \end{array} \quad (1)$$

where $X_{i,j}$ is the value of attribute At_j for UE_i.

Step 2: The values for each attribute have different scales. Moreover, we separate the attributes as desired and undesired attributes based on their impact on overall performance. So, we normalize the decision matrix to make it in uniform scale. The desired attributes are normalize1d as:

$$X_{i,j}^{nor} = \frac{X_{i,j}}{\sum_{i=1}^{m} X_{i,j}} \quad \text{where i} = 1,.\,\text{m and j } 1,\dots\text{p} \quad (2)$$

while the undesired attributes are normalized as:

$$X_{i,j}^{nor} = 1 - \frac{X_{i,j}}{\sum_{i=1}^{m} X_{i,j}} \quad where\ i = 1,.\,m\ and\ j = 1,.\,p \quad (3)$$

 ISBN: 978-1-61208-633-0

Step 3: Then based on impact on the performance we assign weights W to each attribute as:

$$X_{w(i,j)} = W_i * X_{nor} \quad where \ \sum_{i=1}^{m} W_i = 1 \qquad (4)$$

Step 4: Next, we find we find the Ideal positive (I^+) for each attribute and the ideal negative solution (I^-) as:

$$I^+ = \{Max\ (X_{W(1,j)}, X_{W(2,j)}, \ldots\ldots\ X_{W(m,j)})\} \quad (5)$$

$$I^- = \{Min\ (X_{W(1,j)}, X_{W(2,j)}, \ldots\ldots\ X_{W(m,j)})\} \quad (6)$$

Step 5: Then using eq. (5) and (6), we calculate the distance of each solution from (I^+) and (I^-) as:

$$DI_i^+ = \sqrt{\sum_{j=1}^{p} (X_{w(i,j)} - I_j^+)^2} \qquad (7)$$

$$DI_i^- = \sqrt{\sum_{j=1}^{p} (X_{w(i,j)} - I_j^-)^2} \qquad (8)$$

Step 6: We then use DI_i^+ and DI_i^- from eq. (8) and (9) to calculate the coefficient of closeness to the ideal solution as:

$$CC_i = \frac{DI_i^-}{DI_i^- + DI_i^+} \quad where\ i = 1,,m \ while\ 0 \leq CC_i \leq 1 \qquad (9)$$

Step 7: The final step is to rank the UEs in descending order based on the value of CC_i. The UE with highest value of CC_i is the best choice for D2D communication.

III. Simulation And Results

We performed simulations of the proposed scheme in MATLAB considering a small-cell environment with 10 UEs distributed uniformly. Social attributes like frequency of contact, contact duration, social media interaction and similarity of interest are considered as in [4]. Similarly, the physical attributes taken into account include the number of available services, distance from the source UE and the available interfaces. The SCBS provides assistance in D2D peer discovery and is assumed to have the mentioned information about the available UEs.

Initially, each attribute of a UE is assigned random values between 0 and 6. For simplicity, uniform weights are assigned to the attributes after normalization. Three simulation scenarios are considered separately based only on physical, social and both the physical and social attributes.

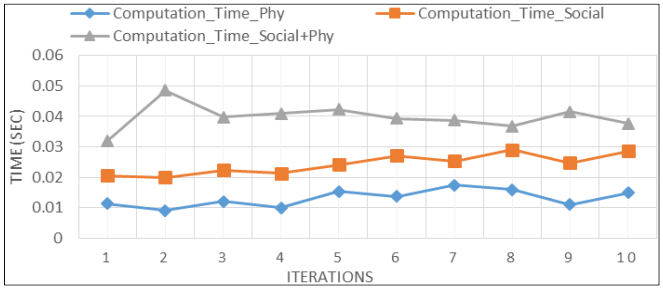


Figure 1. D2D peer Discovery Time vs. Iterations

The simulations are run for ten times and the results are plotted in terms of D2D peer discovery time and Peer discovery success ratio as shown in Figure 1 and Figure 2.

The results in Figure 1 show the D2D peer discovery time for scenario considering physical, social and the combination of physical and social attributes. The average time of discovering the best D2D peer is 39 ms.

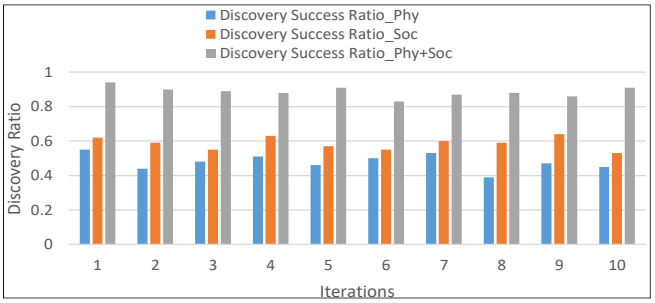


Figure 2. D2D Peer Discovery Success Ratio vs. Iterations

Similarly, we plot the results of D2D peer discovery success ratio based on comparison with the analytical results as shown in Figure 2. The average success ratio in the scenario considering both social and physical attributes is 88%. The proposed method is fast as it takes seven easy steps to find the best D2D peer. Similarly, it efficiently combines attributes of different impact in a simple manner.

IV. Conclusion And Future Direction

In this paper, we proposed a D2D peer discovery method based on multi-attribute decision modeling. The method is simple to implement, fast and improves the peer discovery process significantly. Simulation results demonstrate the efficiency of the method in terms of discovery time and success ratio. In the future, we aim to improve the work by considering more realistic scenarios and determine the impact of each attribute.

References

[1] S. Kim, "D2D-Enabled Small Cell Network Control Scheme Based on the Dynamic Stackelberg Game," Mobile Information Systems, vol. 2017, pp. 1-11, 2017.

[2] D. Tsolkas, N. Passas, and L. Merakos, "Device discovery in LTE networks: A radio access perspective," Computer Networks, Vol. 106, pp. 245-259, 2016.

[3] C. L. Hwang, K.Yoon, "Methods for Information on Attribute Given. Multiple Attribute Decision Making Methods and Applications: A State-of-the-Art Survey," Springer-Verlag Berlin Heidelberg New York: pp. 128-140, 1981.

[4] Z. Zhang, T. Zeng, X. Yu, and S. Sun, "Social-aware D2D Pairing for Cooperative Video Transmission Using Matching Theory," Mobile Networks and Applications, pp. 1-11, 2017.

 ISBN: 978-1-61208-633-0

Telco-Carrier grade Evaluation of 5G SDN/NFV-enabled Policy Control Concepts

Michael Maruschke
Hochschule für Telekommunikation Leipzig (HfTL)
Leipzig, Germany
e-mail: maruschke@hft-leipzig.de

Jan Kasimir and Manuel Keipert
Telekom Deutschland GmbH
Bonn, Germany
e-mail: jan.kasimir@telekom.de
e-mail: manuel.keipert@telekom.de

***Abstract*—The main question that this paper intends to answer is what telecommunication operators could do for their mobile network evolution until 5th Generation (5G) mobile networks are deployed for mass-market and beyond 2020. Especially in scope is policy-related functionality in the core network. The roadmap for mobile networks should take into account technologies such as Software Defined Networks (SDN) and Network Function Virtualization (NFV) but also consider commercial necessities. This paper examines five approaches proposed by the authors, covering the full range from extending the policy controller's capabilities to removing as many elements of the policy framework as possible. Approaches are rated through evaluation criteria chosen by the authors to create a transparent, objective and Telco-Carrier grade feasible conclusion. The conclusion gives operators possibilities for their strategic roadmap towards 5G networks and support any upcoming design decisions.**

Keywords-5G; NFV; PCF; QoS; SDN

I. INTRODUCTION

With the advent of 5G, SDN and NFV, the motivation for a Telco-Carrier to evaluate policy control concepts is paramount for strategic cost and architecture resource planning. However, up to the present, there is no SDN- and NFV-enabled 5G network available for a Telco-Carrier grade deployment of large scale mass-market production.

In infrastructure-owning operators' cellular mobile networks, the policy controller is an essential function. It provides sophisticated policy handling to ensure service quality and charging [1]. It is also required to realize public voice telephony services in packet-switched networks like Voice over LTE (VoLTE) [2]. The policy controller is part of the comprehensive Policy & Charging Control (PCC) framework, consisting also of policy enforcement functions.

The PCC framework in the cellular mobile packet network will undergo significant changes in multiple dimensions with the transition to 5G networks. Until 5G is available and deployed on a large scale, many other technologies impact policy evolution such as SDN and NFV. The complete core network infrastructure is impacted by the emergence of SDN, NFV and 5G. All three technology trends influence different layers of the Telco operator production network, comprising network layer SDN, system virtualization and 'cloudification' NFV up to the 5G application layer.

Beside those big trends, there are many other aspects directly impacting the policy controller. Accordingly, there are use cases (e.g., full flat rate or Internet of Things (IoT) based), where dynamic policy control will be dispensable.

Telco operators are constantly adapting their network strategy to keep up, firstly with a fast-growing number of terminal devices of all kind (Smartphones, tablets, IoT devices, autonomous driving vehicles, etc.), secondly with the rapidly increasing mobile data traffic those devices produce and thirdly with the complexity increases due to new services. Mobile data traffic grew tremendously in the last few years and forecasts indicate that this trend will continue in the future [3]. The diversity of terminals and use cases that will come with IoT and 5G will put even more demanding requirements on the Telco network infrastructure, for example in terms of latency, bandwidth and other Quality of Service (QoS) related parameters [4]. The PCC framework is key for dynamic QoS policing.

With regards to the PCC framework that provides data charging, VoLTE enablement and QoS handling for data sessions, the evolution towards a 5G network implies many challenges. The diversity of use cases and its usage patterns changes profoundly from 4th Generation (4G) to 5G networks [4]:

- QoS functionalities will be refined to fulfill stringent 5G use case and become highly adaptive (dynamic real-time adapting).
- Other new use cases like Vehicular-to-Everything (V2X) and IoT might not require a dynamic policy controller at all. This QoS-related challenge in 5G of predictable end-to-end (E2E) latency of a few milliseconds might not utilize a centralized Policy & Charging Rules Function (PCRF) in the mobile core network.
- Static versus dynamic rule assignment in 5G will depend on the use case. Static rule assignment bears cost-saving potential as no full-fledged PCRF is required.
- Decentralization of control functions to the core edge (assuming that this is possible) enables the network to cope with strict latency requirements.

 ISBN: 978-1-61208-633-0

In the process of SDN/NFV-based 5G network transformation, there are opportunities for architecture design improvement that have been identified by the authors. This paper presents five approaches that propose adaptations to the Policy Control Function (PCF) – the policy controller in 5G.

In section II, related work and a standardized context of basic architectures for the 5G network, SDN and NFV are introduced. Section III covers the descriptions of the five approaches. In section IV the approaches are evaluated and compared, resulting in our recommendations for Telco operators as to how to include these into their roadmaps. Section V concludes with a summary of the next steps towards 5G for Telco operators.

II. RELATED WORK

A. Research Contributions

In 2015, an academic study was focusing on a fair QoS resource reservation concept for upcoming 5G networks [5]. Nevertheless, this work was not integrated into the 3GPP 5G specifications.

Two more recent contributions address the issue of resource allocation for a key element of 5G networks, the network slices. One paper introduces three new network functions to analyze traffic per network slice, to manage the access rights on network slice resources, and to provide an adaptive traffic forecast model [6]. The other paper presents a model to orchestrate slices based on typical service demands considering the underlying resource capacities [7]. Both research papers relate to optimization of network slice resource usage. Nevertheless, those are pure simulations or calculations that were not verified under real Telco-Carrier network conditions. Furthermore, all three contributions do not address the functional core network nodes that are introduced with 5G, e.g. PCF or User Plane Function (UPF).

B. 5G/SDN/NFV Reference Architectures

The approaches - presented in section III - are based upon 5G/SDN/NFV architectures. For a better understanding, these basic architecture designs are presented in this section.

With regards to 5G, the 3GPP Release 15 proposed system architecture in a domestic scenario is used [8]. Figure 1 shows this 3GPP 5G architecture with service-based interfaces within the control plane.

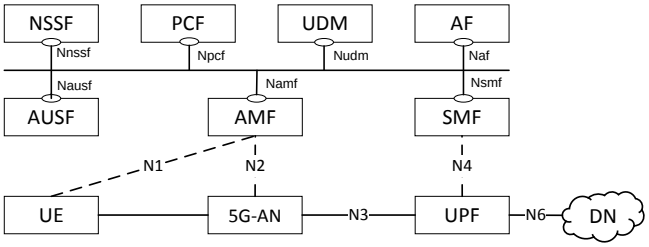


Figure 1: Service-based 3GPP 5G Architecture (cf. [8])

Besides other logical network functions, the control plane includes the PCF and the Session Management Function (SMF). The SMF operates as centralized element between logical control plane network functions and the UPF via interface N4.

A high-level SDN architecture, taken from the ITU-T Recommendation Y.3300 [9], is shown in Figure 2. Stimulated from SDN applications, the centralized SDN controller manages the local existing network devices (in the SDN Resource Layer).

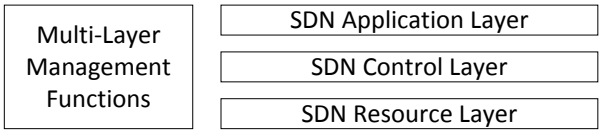


Figure 2: SDN Architecture in three Layers (cf. [9])

To operate and provide a NFV environment with all required configurations, a Management and Orchestration (MANO) framework is specified. The European Telecommunications Standards Institute (ETSI) determined a high-level functional architectural framework for NFV and MANO [10].

Figure 3 displays the ETSI NFV MANO architecture comprising the three functional nodes NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtualized Infrastructure Manager (VIM).

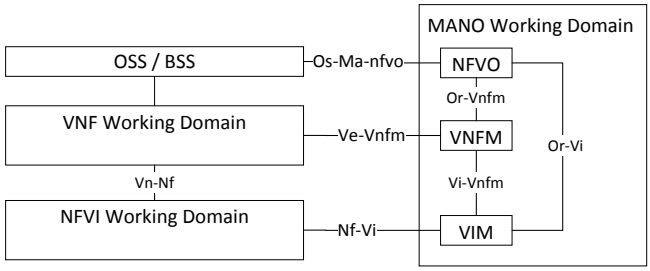


Figure 3: NFV MANO Architecture (cf. [10])

III. PCC APPROACHES

The goal of all approaches is to either extend the policy controller's capabilities to improve the network, or remove as many elements of the policy framework as possible in order to reduce complexity and costs.

A. SMF-integrated PCF

In this approach, the PCF is integrated into the SMF to optimize the logical infrastructure. The resulting architecture is depicted in Figure 4 that shows an SMF with policy control functionalities.

The resulting architecture is more compact. The extended SMF (enhanced with PCF) unites session management functions (manage session establishment, modification and release; control Internet Protocol (IP) address allocation; advice policy enforcement in UPF) and the policy management (evaluate policy decisions).

 ISBN: 978-1-61208-633-0

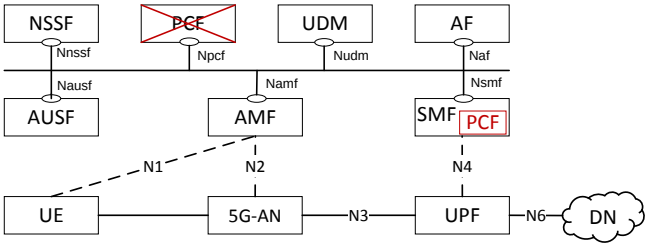


Figure 4: Overview for SMF-integrated PCF Approach

The service-based interface Npcf between a separated PCF and SMF can be omitted. Interactions between PCF and other required functionalities (e.g. Application Function (AF), Unified Data Management (UDM)) corresponding to the 3GPP TS 23.503 (section 5.3) [11] will be realized via SMF.

B. SDN-based PCF

In this approach, a new interface between SMF and SDN controller is introduced exchanging information between PCF and SDN controller over SMF. This enables the SDN controller to decide whether and how to include such information into its transport network steering process.

The precondition is that the mobile core network is deployed on top of an SDN-controlled network infrastructure.

As it can be seen from Figure 5, mobile core network functions operate on the application layer from an SDN architectural point of view. The SDN controller processes information provided by SMF. The decision for a SMF (and not PCF) to SDN controller interface is justified through the SMF's holistic view of the data sessions. The SDN controller steers the network device in the resource layer. Management functions are outlined on the left side of the figure.

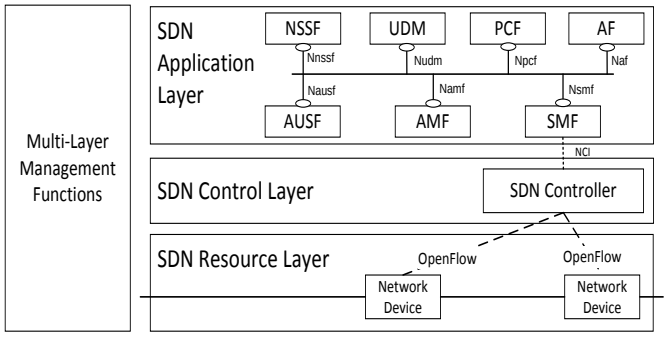


Figure 5: Overview for SDN-based PCF Approach

The SMF delivers session/subscriber related information from PCF via the SDN controller's Northbound Controller Interface (NCI). On this way, the SDN controller becomes service aware and is enabled to adapt the underlying transport layer forwarding path and capacity for a data flow according to a service's requirements, e.g., latency and QoS.

C. Slice-specific PCF

This approach proposes multiple instances of slice-specific PCFs (sPCF). The sPCFs are responsible for one single slice, tailored to serve the slice-specific use cases.

The precondition for this approach is Network Slicing. When a subscriber establishes a new session, the Network Slice Selection Function (NSSF) assigns the session to a slice. A common PCF (cPCF) is deployed in a common domain and coordinates policy management. It is connected to the NSSF and provides relevant information to the slice that is in charge of the subscriber.

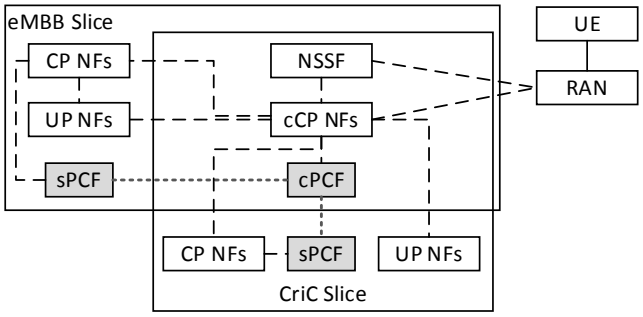


Figure 6: Overview for Slice-specific PCF Approach (cf. [12] figure 6.1.1.1)

Figure 6 shows an operator's mobile core network with two slices and one common domain in the middle – the area where the two slices overlap (cf. [12] figure 6.1.1.1) . Both slices, enhanced Mobile Broadband (eMBB) and Critical Communications (CriC), have their own sPCF that is connected to the cPCF in the common domain. Other common Control Plane Network Functions (cCP NFs) can be found in the common domain. Furthermore, there are dedicated Control and User Plane Network Functions (CP NFs and UP NFs) in each specific slice. The cPCF is approached first when a customer session establishment request is sent. It is connected to all sPCFs via an interface and forwards the request to the most suitable sPCF instance in a slice. The Radio Access Network (RAN) is informed about the chosen slice and forwards subsequent control and user plane traffic directly to the slice's functions.

D. PCF-based NFV orchestration

In this approach, a virtualized PCF is connected to the VNFM function in the NFV MANO working domain, depicted in Figure 7.

A precondition is that the 5G reference architecture is virtualized in an NFV environment. PCF and other network functions are deployed as VNFs.

The virtualized PCF delivers aggregated application layer information about subscribers, services used, etc., in real-time via Operations Support Systems/ Business System Support (OSS/BSS) layer and NFVO to VNFM for a more accurate VNF lifecycle management and monitoring purposes.

 ISBN: 978-1-61208-633-0

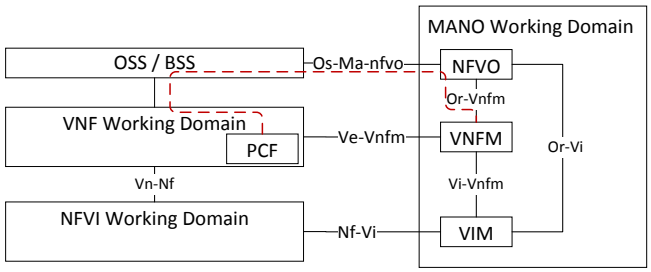


Figure 7: Overview for Virtualized PCF Approach in NFV MANO

PCF can provide, for example, the type of device, type of service or device location. Such information allows the VNFM to handle VNF lifecycle management (VNF instantiation, scaling and termination) more accurately with regards to the demand, quality requirements, and so forth.

E. Obsolete PCF

There are multiple use cases already today and there will be even more in the future that do not require a PCF. Typically, operators aim for network simplicity, hence in today's network the PCRF often controls all mass market subscribers. This approach recommends not applying (dynamic) policy control for some new IoT and 5G use cases as well as existing use cases where dynamic PCC is not needed. Hence a valid strategy option is to remove customers and use cases from the PCF.

IV. Approach Evaluation

Several evaluation criteria have been created, particularly from an infrastructure-owning Telco-Carrier perspective. The criteria are a mixture of standard conformity and technical design aspects, as well as commercial considerations.

A summary of the evaluation of our five PCC approaches is presented in a table.

Finally, the approach rating is illustrated more detailed.

A. Definition of Evaluation Criteria

1) Standards Developing Organization (SDO) conformity (SDO conf.)

This criterion verifies the conformity with regards to the existing technical standards (e.g., 3GPP specifications).

2) Pre-5G compliance (Pre-5G)

This criterion evaluates whether an approach could be implemented in a 4G network.

3) Complexity of Integration and Operation (Compl.)

This criterion evaluates the complexity of the technical implementation and operation for the approach.

4) Cost (Cost)

This criterion estimates the incurred costs for the approach. It excludes any costs for SDN and NFV. Here we assume that costs include Capital Expenditure (CAPEX) and Operational Expenditure (OPEX).

5) Consumer Business Opportunities (Business)

This criterion estimates if new business opportunities are generated by this approach, with regards to additional revenue.

B. Summary of the Evaluation

Table 1 presents our approach evaluation summary. The evaluation is done via a three-tier scale. The scale is defined as follows:

- the approach has a positive effect with regards to the evaluation criterion: '+'
- the approach has a neutral influence with regards to the evaluation criterion: 'o'
- the approach has a negative effect with regards to the evaluation criterion: '-'.

Table 1: Approach Evaluation Summary

Evaluation criteria	Approaches				
	SMF-integrated PCF	SDN-based PCF	Slice-specific PCF	PCF-based NFV orchestration	Obsolete PCF
SDO conformity	o	+	+	+	-
Pre-5G-compliance	-	+	+	+	+
Complexity of Integration and Operation	-	o	o	o	+
Costs	+	o	-	-	+
Consumer Business Opportunities	o	+	+	o	-

C. PCC Approach Rating

1) SMF-integrated PCF

The approach centralizes policy and session management in one network node and it is assessed by the evaluation criteria as follows:

a) *SDO conf.: PCF interactions with other control plane functions (e.g., UDM) will be integrated into SMF*

b) *Pre-5G: Evolved Packet Core (EPC) point-to-point interface design causes unmanagable complexity*

c) *Compl.: shared PCF/SMF interface complicates maintainability*

d) *Cost: no additional operational costs for dedicated PCF; further savings through licensing cost reduction*

e) *Business: no new business opportunity, but existing use cases remain*

2) SDN-based PCF

The approach enables mobile data session information exchange to an SDN controller and it is assessed by the evaluation criteria as follows:

a) *SDO conf.: no fundamental change in design principles*

b) *Pre-5G: only requirement is the deployment of a SDN*

c) *Compl.: complication through NCI design & integration effort*

d) *Cost: minor interface design and integration costs*

e) *Business: new service quality enforcement options on network layer, although not visible to end customer*

3) Slice-specific PCF

The approach decentralizes policy control functionality and it is assessed by the evaluation criteria as follows:

a) *SDO conf.: no fundamental change in design principles*

b) *Pre-5G: requirements are deployment of NFV and Network Slicing*

c) *Compl.: decentralization facilitates PCF with reduced functionality, but interactions between PCFs drives complexity*

d) *Cost: increase due to acquisition and operation for multiple PCF nodes*

e) *Business: offer services for diverse use cases in dedicated independent slices*

4) PCF-based NFV orchestration

The approach improves NFV MANO processes and it is assessed by the evaluation criteria as follows:

a) *SDO conf.: no change in design principles*

b) *Pre-5G: only requirement is the deployment of a NFV*

c) *Compl.: complication through new interface design & integration effort*

d) *Cost: minor interface design and integration costs; MANO function enhancement to incorporate interface*

e) *Business: no new business opportunity, but existing use cases remain*

5) Obsolete PCF

The approach reduces complexity in the mobile core network and it is assessed by the evaluation criteria as follows:

a) *SDO conf.: PCF removal is not foreseen by 3GPP*

b) *Pre-5G: same constraint as for 5G, namely no more dynamic policy handling*

c) *Compl.: no PCF node and streamlining of use cases*

d) *Cost: neither licensing nor operational costs for PCF*

e) *Business: reduction of feasible use cases*

V. CONCLUSION

The best improvement for the pre-5G and 5G architecture in the authors' view is a combination of the approaches depending on the use cases. This is most easily done in a sliceable, virtualized and programmable core network, which is expected to be built in the near future to be prepared for 5G.

A. Combining Approaches

In 5G, the recommended basic setup is to have one slice per 3GPP use case category [13]: eMBB, CriC, Massive Machine Type Communications (mMTC), Enhancement of Vehicle-to-Everything (eV2X).

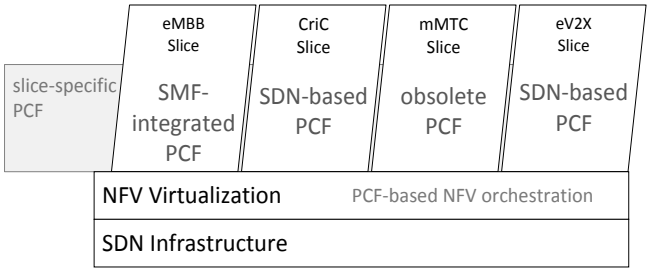


Figure 8: Combining Approaches

Figure 8 constitutes a sliced operator network on top of an NFV-virtualized and SDN-controlled infrastructure, where all approaches are implemented. The slice-specific PCF approach is implemented implicitly, as Network Slicing is employed and multiple PCFs are used. The PCF-based NFV orchestration approach is also implied as each slice

 ISBN: 978-1-61208-633-0

PCF can integrate the interface to VNFM if required. For each slice, there is one approach applied:

a) eMBB slice

Operates conventional mobile data use cases (that require high bandwidth) and VoLTE. The SMF-integrated PCF approach is advisable here, especially with respect to VoLTE.

b) CriC slice

Use cases in this category are for example factory automation and telemedicine. Low latency and high availability is essential to such critical communication. The network can support fulfillment of such demands through optimized routing with the help of an SDN. Applying the SDN-based PCF approach further improves the route optimization process of the SDN controller.

c) mMTC slice

The focus in this slice lies on IoT use cases that need high resource and energy efficiency. Wearables, sensors and other IoT devices rely on a long-lasting power supply and transmit typically very small data. Dynamic policy rule evaluation is not required; hence the obsolete PCF approach is recommended to reduce complexity, latency and save money.

d) eV2X slice

Use cases demands are similar to CriC use case demands (low latency and high availability) plus safety aspects and positioning accuracy. Therefore, the SDN-based PCF approach is a suitable option.

B. Final considerations

It should be stated that there is no 'one-size-fits-all approach' that is equally suitable for all types of operators. Moreover, there is no single approach for an operator but always a well-balanced mixture to bridge the time between 4G deployments and future SDN and NFV based 5G networks. Operators are diverse in terms of technical, infrastructural and financial aspects as well as service offerings, market position, etc. Those aspects matter as these drive the complexity of the network, which makes it usually difficult for big infrastructure-owning Telco-Carriers to adapt. All results and recommendations must be interpreted by the operators. Combining these approaches to the operators needs provide a real benefit partially already now as well as for sure with 5G based on the flexibility of a SDN/NFV Telco cloud.

Beside these overarching strategic results, the paper proposes five PCC approaches for network enabling and significant cost cutting.

This paper is intended to allow the operator to adapt the 3GPP 5G reference architecture to meet its carrier-specific needs and network strategy. Nevertheless, for each operator the challenge remains to develop its real deployed carrier networks and processes towards a 5G-enabled policy control based network.

REFERENCES

[1] 3GPP, TS 23.203 - Policy and Charging Control Architecture (Release 15) V15.2.0 (2018-03).

[2] GSM Association, Official Document IR.92 - IMS Profile for Voice and SMS V11.0, 2017. Available: https://www.gsma.com/newsroom/wp-content/uploads//IR.92-v11.0.pdf.

[3] Ericsson, June 2017. [Online]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.

[4] 3GPP, TS 22.261 - Service requirements for the 5G system V16.2.0 (2017-12).

[5] R. Trivisonno, R. Guerzoni, I. Vaishnavi und A. Frimpong, „Network Resource Management and QoS in SDN-Enabled 5G Systems," 2015 IEEE GLOBECOM , Dec. 2015, ISBN: 978-1-4673-9526-7.

[6] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia und A. Banchs, „Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," IEEE Conference on Computer Communications (INFOCOM 2017), May 2017, ISBN: 978-1-5090-5336-0.

[7] D. T. Hoang, D. Niyato, P. Wang, A. De Domenico und E. Calvanese Strinati, „Optimal Cross Slice Orchestration for 5G Mobile Services," 16 December 2017. submitted [Online]. Available: https://arxiv.org/abs/1712.05912.

[8] 3GPP, TS 23.501 - System Architecture for the 5G System (Release 15) V15.0.0 (2017-12).

[9] ITU-T, Recommendation Y.3300 - Framework of software-defined networking, (06/2014).

[10] ETSI, GS NFV-MAN 001 – Network Functions Virtualization (NFV); Management and Orchestration v1.1.1, (2014-12) .

[11] 3GPP, TS 23.503 - Policy and Charging Control Framework for the 5G System (Release 15) V15.0.0 (2017-12).

[12] 3GPP, TR 23.799 - Study on Architecture for Next Generation System (Release 14) V14.0.0 (2016-12).

[13] 3GPP, TR 22.891 - Feasibility Study on New Services and Markets Technology Enablers (Release 14) V14.2.0 (2016-09).

 ISBN: 978-1-61208-633-0

Towards a Composition of Region-Adherent Systems

Dilshod Rahmatov, Manuel Gieseking, Oliver Theel
Carl von Ossietzky University of Oldenburg
Department of Computer Science
Oldenburg, Germany
Email: dilshod.rahmatov@informatik.uni-oldenburg.de

***Abstract*—Graceful degradation entails a loss of functionality or reduction in the service quality that a system provides in response to faults. One new form of graceful degradation, which is called region adherence, upper-bounds the violation of safety due to faults *in space*. Bounding in space means that the decrease of service quality, which the system provides to its environment, is upper-bounded per fault. In this paper, we investigate the effect of composing region-adherent systems such that the resulting system is also region-adherent. We analyze the service quality of region-adherent wireless sensor networks, which are structured according to different topologies. We cover all possible fault scenarios and analyze their impact on differently structured networks by calculating the minimal, the average, the standard deviation, and the variance of the resulting service qualities.**

Keywords–Wireless Sensor Networks; Service Quality; Fault Tolerance; Graceful Degradation; Region Adherence.

I. INTRODUCTION

Nowadays, the increasing use of computers in almost every field of modern life has led to a need for highly dependable computer systems. Fault tolerance is a well-known approach by which the dependability of computer systems can be increased. [1] The goal of fault tolerance is to provide the intended service of a system despite the presence of faults in the system. Graceful degradation [2] is one of the possible concepts to realize a fault-tolerant system. In graceful degradation, the execution of the system continues but in a "degraded" mode. A "degraded" mode is a limited functionality or performance of the system even when a large portion of the system has been destroyed or rendered inoperative.

Region adherence [3] is a particular form of graceful degradation: a system may be implemented based on a distributed algorithm that employs redundancy to bound the observable, incorrect behavior of the system *in space*. With "bounding in space," it is meant that the decrease of the quality of the service, which the system provides to its environment, is upper-bounded per fault. We call such an algorithm (resp. the system implementing it) *region-adherent*. Let f be the maximal number of faults a system is able to withstand. Then, a region-adherent system – per fault – gracefully degrades the service quality provided by the system. It does so up to some maximal number of faults. Additionally, degradation is upper-bounded per fault. A region-adherent system exhibits very desirable fault tolerance properties as it provides *quantified service quality guarantees* in case up to f faults happen. Thus, at any time, knowing the number of faults that already have occured, the system user can take an *a priori*-known minimal service quality for granted: a very valuable information in various critical settings. Examples of region adherent systems are presented in [3] and [4] along with a formal definition of region adherence and formal techniques for proving the region adherence property. Becker et. al. [3] introduced the concept of region adherence and presented a Wireless Sensor Network (WSN) for monitoring the air humidity in some geographical region. In this WSN, a region-adherent algorithm is used that helps to quantify the effects of failed sensors in the sensing process of the WSN. A simple refinement of the lower bounds of the service quality known of a region-adherent system is presented in [5].

The main idea of this paper is to compose region-adherent systems out of region-adherent subsystems and analyze the impact of faults on the service quality of the resulting (region-adherent) system. For this purpose, we investigate five tree-shaped WSN topologies, each with nine sensor nodes subject to faults. We show that, for these particular examples, the parallel composition of the subsystems lead to a higher average of service quality than composing the subsystems in hierarchical fashions. Furthermore, we gain the best results by distributing the sensor nodes between the parallel subsystems as equally as possible.

The paper is structured as follows. In the next section, we briefly introduce the notion as well as a formal definition of region adherence together with closely related notions needed for the understanding of the presented material. In Section III, we give a general formula for calculating the average service quality of a region-adherent tree-shaped WSN. Section IV analyses five compositions of region-adherent WSNs based on tree-shaped topologies with nine sensor nodes each using the general formula. In Section V, we summarize the results of the paper.

II. NOTION OF REGION ADHERENCE

The basic idea of the region adherence concept is to restrict the safety property invalidation of a system *in space*. This is achieved by upper-bounding the reduction of the service quality of a system *per fault up to some maximal number of faults*. It is important to note that region-adherent systems can only by realized, if the possible faults which might occur (i.e., the faults of the underlying fault model) may not transfer a system from a given state to *any other state* in the system state space.

For an example of the fault-tolerant behavior of a region-adherent system, refer to Figure 1. The dashed blue line shows the service quality of the region-adherent system in a particular execution of the system. The solid blue line (i.e., the "staircase-like graph") states the *minimal* service quality guaranteed by the system. Irrespective of a particular execution, the system's service quality *always* remains on or above the minimal service quality represented by the "staircase-like graph." In the example, the system is initially in a safe state. In a safe state, it always delivers 100% of service quality. After the first

 ISBN: 978-1-61208-633-0

fault has occurred, the system must deliver a service quality of at least $100\% - \alpha$. Note that the dashed line actually exhibits a slightly higher service quality. This is possible for a particular execution but may not generally hold. After the second fault, the guaranteed and delivered service quality is again reduced, but it becomes not lower than $100\% - \alpha - \beta$. After the third fault, the guaranteed service quality is at least $100\% - \alpha - \beta - \gamma$.

Figure 2 gives a topological interpretation of a region-

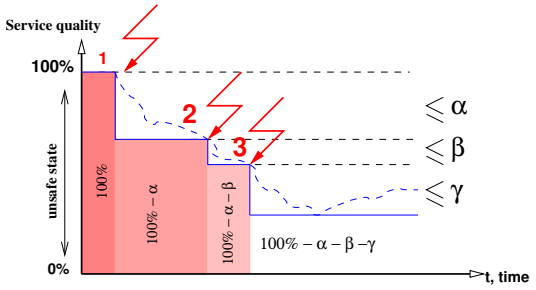


Figure 1. Worst case (solid blue line) and particular behavior (dashed blue line) of a region-adherent system after one, two, and three faults over time with $\alpha, \beta, \gamma > 0$.

adherent system: here, the blue dashed line, again, represents a particular execution of the region-adherent system over time. The execution starts in the innermost region where the system exhibits 100% of service quality. When the first fault occurs, the system is thrown into a state of the neighboring region with only $100\% - \alpha$ service quality. The next fault lets the system enter a region with only $100\% - \alpha - \beta$ service quality and so on. When being in a particular region – without any further fault occuring – the system is allowed to stay within the region, it is currently in, as well as in any included region. In this sense, the system behavior is *restricted in space*, i.e., it must *adhere to regions* of known system quality. We formally

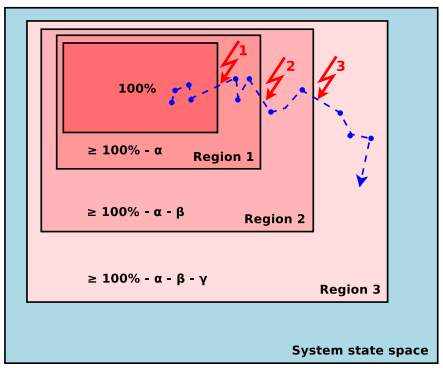


Figure 2. Topological interpretation of a region-adherent system with $\alpha, \beta, \gamma > 0$. The dashed blue line represents a particular execution of the system where three faults occur.

define the notation of region adherence as follows [6]:

Definition 1 (General Region Adherence of a System): We assume a system with configurations C, initial configurations C_0 and algorithm A under fault model F. Let $g : C \mapsto [0,1]$ be a function stating the service quality of the system and let f be a natural number. $r : \{0, \ldots, f\} \mapsto [0,1)$ is a non-decreasing function with $r(0) = 0$ and $r(f) < 1$.

Algorithm A is called *f-region-adherent wrt. g, r, and F*, if and only if for all reachable configurations $c \in C$, all initial configurations $c_0 \in C_0$ and all executions $\gamma = c_0 \cdots c$ ending in c with $\#_{F \setminus A}(\gamma) \leq f$, the following holds:

$$g(c) \geq 1 - r\left(\#_{F \setminus A}(\gamma)\right), \tag{1}$$

where $\#_{F \setminus A}(\gamma)$ represents the number of fault steps of execution γ.

Figure 3 gives an example of an execution γ of the system. In this execution, three faults occur. Thus, $\#_{F \setminus A}(\gamma)$ is 3. A

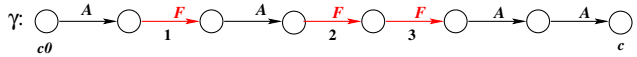


Figure 3. Example of an execution γ.

system executing an f-region-adherent algorithm is also called f-region-adherent. The value of g may be interpreted as a percentage. The function r can be perceived as the service's *loss* or *reduction of quality* with $r(i)$, $0 \leq i \leq f$, upper-bounding the loss due to the i-th fault. Note that an f-region-adherent system is able to tolerate at least f faults and is still exhibiting a service quality higher than 0%.

A particular subclass of region-adherent algorithms is called *α-equidistance*. It allows the classification of region adherence based on a simple service quality reduction function r. Furthermore, algorithms can often more easily be proven α-equidistant than region-adherent in general. This is due to the proof scheme adopted (see Becker et. al. [4]).

Definition 2 (α-Equidistance): An algorithm A that is f-region-adherent wrt. g, r, and F with

$$r(i) = \alpha \cdot i, \qquad 0 \leq i \leq f, \tag{2}$$

is called *α-equidistant f-region-adherent*.

For an α-equidistant algorithm, $1/f > \alpha > 0$ trivially holds.

In the next section, we present a formula for calculating the service quality of a tree-shaped sensor network. This formula is the extension of a formula presented in [3] for the arithmetic mean of sensor values to sensor systems which are composed of different subsystems. We use this formula for the calculation of the service quality of region-adherent WSNs presented in Section IV.

III. Calculation of the Service Quality of Region-Adherent Tree Topology Networks

The systems under consideration consist of gateway nodes (also called *sink* nodes), which collect the data of all connected *sensor* nodes. Sensor nodes produce or sense the data, the gateway nodes as well as the system users are interested in. Gateway and sensor nodes form a (communication) network. This network is of a tree topology. Sensor nodes form the leafs of the tree topology and the only or the multiple gateway nodes form the root and the inner nodes (if any) of the tree topology. We call such systems *sensing tree networks*.

Since gateway nodes, in our setting, are realized by much more dependable hardware components, gateway nodes are assumed to be failure-free. Sensor nodes, on the contrary, are realized by cheap, unreliable hardware. They are assumed to be subject to faults leading to failures. If a sensor node fails, then we assume a fail-fast failure semantics. In subsequent figures, sink nodes are depicted as black circles whereas sensor nodes are represented by white circles.

 ISBN: 978-1-61208-633-0

In the following, we define a sensing tree network formally.

Definition 3 (Sensing Tree Network (with Failures)): A tuple $S = (N, B, W, T, b)$ with a set of *nodes* N, a set of *gateway* nodes $B \subseteq N$, a set of *sensor* nodes $W \subseteq N$, a *transition relation* $T \subseteq B \times N$, and an *initial* node $b \in B$ is called *sensing tree network*. For T, we demand two constraints. Firstly, for every two nodes $n_1, n_2 \in N$ with $n_1 \neq n_2$ there must exist exactly one path between n_1 and n_2. Secondly, for every $b \in B$ there must be a node n with $(b, n) \in T$.

For a set of *faulty nodes* $F \subseteq W$, we call $S = (N, B, W, T, b, F)$ a *sensing tree network with failures*.

As an example, the sensor network depicted in Figure 4 is defined as $S_A = (\{s_0, s_1, \ldots, s_9\}, \{s_0\}, \{s_1, \ldots, s_9\}, \{(s_0, s_i) \mid s_i \in W\}, s_0)$, with s_0 as the given root gateway node.

In our setting, the sensors are distributed in a (small) geographical region, but should sense redundantly the same data.

For a given sensing tree network with failures $S = (N, B, W, T, b, F)$, we assume that each sensing node $w \in W$ delivers a value $v(w) \in [0, u]$ for a given bound u. Let V denote the exact value which should have been sensed, then each correctly sensed value can deviate from this value by at most $\epsilon > 0$. Since we only want to investigate the impact of a faulty node on the service quality in different topologies, we abstract from drawing the value of a faulty node randomly but consider its value to be zero. Thus,

$$v(w) \in \begin{cases} [0,0] & \text{if } w \in F \\ [V-\epsilon, V+\epsilon] \cap [0,u] & \text{otherwise} \end{cases}$$

For collecting all child nodes of a given gateway node, we define the function $c : B \to \mathbb{P}(N)$ with $c(b) = \{n \in N \mid (b, n) \in T\}$.

The *mean value of a gateway node* is calculated by $\tilde{v} : B \to \mathbb{Q}$ with

$$\tilde{v}(b) = \frac{1}{|c(b)|} \cdot \left(\sum_{w \in c(b) \cap W} v(w) + \sum_{b' \in c(b) \cap B} \tilde{v}(b') \right). \quad (3)$$

Since all sensors should sense the same value, we estimate their value by calculating the arithmetic mean. The left summand calculates the sum of the values of the sensor nodes directly connected to the gateway node. If a gateway node has other gateway nodes as children by itself, we calculate their value recursively and sum them up in the right summand. The mean value of a gateway node is this sum divided by the number of all children.

We now calculate the *service quality of a sensing tree network with failures* $S = (N, B, W, T, b, F)$ analogously to the approach presented by Becker et. al. [3] for only one gateway node:

$$g(S) = min \left\{ 1 - \frac{|\tilde{v}(b) - V| - \epsilon}{u}, 1 \right\} \quad (4)$$

For comparing the impact of failures on differently structured sensing tree networks, we calculate the mean of all possible occurrences of failures (i.e., failure scenarios). For a sensor tree network $S = (N, B, W, T, b)$ and f failures, we define the set of all possible sensing tree networks with failures as $SF(S, f) = \{(N, B, W, T, b, F) \mid F \subseteq W \wedge |F| = f\}$. This means that $SF(S, f)$ contains all possible failure scenarios of a given sensing tree network with a fixed number f of failures. Thus, we can use the binomial coefficient $\binom{|W|}{f}$ to calculate $|SF(S, f)|$, since we consider all distinct subsets with f elements of the set of all sensor nodes W.

The *mean service quality of a sensing tree network* $S = (N, B, W, T, b)$ *with* f *failures* is then calculated by

$$\tilde{g}(S, f) = \sum_{S_F \in SF(S,f)} \frac{g(S_F)}{|SF(S, f)|} \quad (5)$$

This means that we sum up the service quality of a sensing tree network for each possible failure scenario having exactly f failures and divide them by the number of all possible failure scenarios.

The next section exploits these formulas for analyzing five compositions of sensing tree networks by their service quality despite the occurrence of failures.

IV. Simulation and Evaluation of Sensing Tree Networks

As an example, we assume a WSN for measuring the air humidity. The WSN consists of $n = 9$ sensor nodes that independently measure the air humidity in some (small) geographical region. The sensor nodes send the measured values to some gateway node. We make the following assumptions in our simulations and analyses:

1) sensor nodes work independently of each other, they do not communicate with each other and independently send their values to a gateway node,
2) gateway nodes are failure-free,
3) sensor nodes fail independently of each other,
4) failing sensor nodes do so with fail-silent failure semantics.

A. Test Setup:

We compose the WSN in a variety of tree-shaped typologies. For each topology, we simulate faults at all possible positions and for all possible combinations of positions. For all possible combinations of a fixed number of i failures, $0 \leq i \leq 8$, we calculate the service quality of the whole network using Formula 4, assuming a true air humidity value V of 0.6, a deviation $\epsilon = 0.03$ of a sensor node from true air humidity as well as an upper-bound sensor value u of 0.75. (The particular semantics of this air humidity value is not important in the scope of this paper.) To analyze the simulation results, we calculate the minimal, the average, the standard deviation, and the variance for each number of possible faults of sensor nodes.

B. Test Results:

Topology A: The WSN in Figure 4 has nine sensor nodes which form a 2-level tree network with the gateway node as its root. This arrangement represents Topology A. According to Definition 2, the WSN is 8-region-adherent and α-equidistant with $\alpha = 1/9$. In this topology, obviously, all sensor nodes reside on the same level. As said, we consider all possible combinations of sensor node failures and calculate the minimal, the average, the standard deviation, and the variance of the service quality.

 ISBN: 978-1-61208-633-0

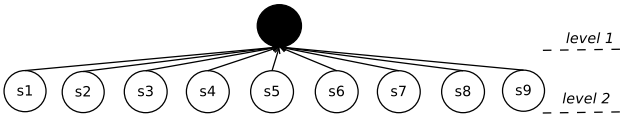


Figure 4. Topology A: WSN with two levels, one gateway node and $n = 9$ sensor nodes. All sensor nodes are on level 2 whereas the gateway node forms the root of the network on level 1.

For example, there are nine failure scenarios where exactly one sensor node fails and 36 failure scenarios with exactly two failures. The results for a WSN using Topology A are given in Table I. For this topology, the standard deviation is

TABLE I. TEST RESULTS OF THE WORST CASE, THE AVERAGE, THE STANDARD DEVIATION, AND THE VARIANCE OF THE SERVICE QUALITY OF A WSN AFTER EACH NUMBER OF FAULTS FOR TOPOLOGY A.

# of Faults	Minimal Service Quality	Average Service Quality	Standard deviation	Variance
0	1.0	1.0	0.0	0.0
1	0.9384	0.9484055471	$3.507 * 10^{-5}$	0.00592
2	0.8318	0.8528330018	$6.590 * 10^{-5}$	0.00811
3	0.7421	0.7570313619	$4.484 * 10^{-5}$	0.00669
4	0.6449	0.6620351405	$3.947 * 10^{-5}$	0.00628
5	0.5516	0.5654450820	$2.983 * 10^{-5}$	0.00546
6	0.4608	0.4719918291	$2.198 * 10^{-5}$	0.00468
7	0.3686	0.3759426946	$1.368 * 10^{-5}$	0.00369
8	0.2770	0.2801876536	$6.472 * 10^{-6}$	0.00254

very small, since all sensor nodes are positioned on the same level within one group. Thus, any subset of sensor nodes failing conceptionally has the same impact on the reduction of service quality. This can more obviously being seen in Figure 5.

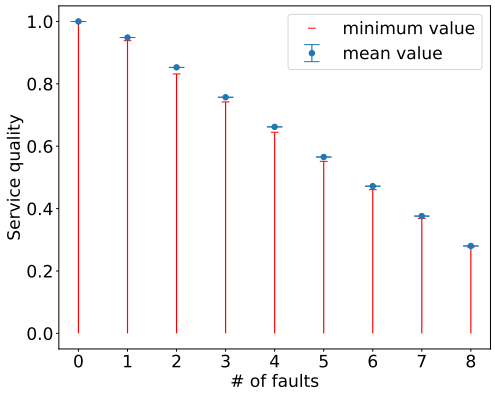


Figure 5. Average and minimal service quality together with standard deviation error bars for a WSN according to Topology A.

Topology B: Here, we consider a 3-level tree topology of the form given in Figure 6. Four sensor nodes together with an additional gateway node are placed on level 2. The additional gateway node manages five sensor nodes on level 3. These five sensor nodes together with their gateway node form a subsystem with a topology similar to Topology A. A WSN using Topology B can be perceived as a WSN that is composed hierarchically by a "parent" system on levels 1 and 2 – which is 3-region-adherent – and a subsystem, which is 4-region-adherent. The entire system obviously is 8-region-adherent.

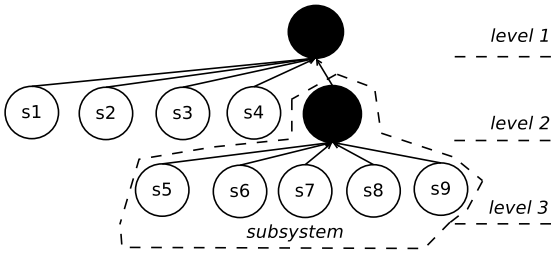


Figure 6. Topology B: WSN with three levels and $n = 9$ sensor nodes. The five sensor nodes on level 3 together with a gateway node on level 2 form a subsystem.

Again, we inject an increasing number of faults into the system and calculate the minimal, the average, the standard deviation, and the variance of the service quality. The results are given in Table II. It can be seen that the possibility of faults occurring

TABLE II. TEST RESULTS OF THE WORST CASE, THE AVERAGE, THE STANDARD DEVIATION, AND THE VARIANCE OF THE SERVICE QUALITY OF A WSN AFTER EACH NUMBER OF FAULTS FOR TOPOLOGY B.

# of Faults	Minimal Service Quality	Average Service Quality	Standard deviation	Variance
0	1.0	1.0	0.0	0.0
1	0.8662	0.9438916178	0.0037	0.06090
2	0.6966	0.8524241592	0.0080	0.08958
3	0.5189	0.7571151734	0.0110	0.10501
4	0.3534	0.6620938031	0.0116	0.10799
5	0.3219	0.5667667375	0.0115	0.10753
6	0.2857	0.4717086893	0.0105	0.10295
7	0.2525	0.3756783898	0.0080	0.08970
8	0.2185	0.2806518074	0.0045	0.06780

on different levels impact the service quality of the system differently, compared to faults in Topology A: faults occurring on a lower level (i.e., level 3) have less an impact on service quality than faults occurring on a higher level (i.e., level 2) of the system. This fact leads to higher standard deviations and variances than in Topology A. This can also be seen in Figure 7. When comparing the minimal service qualities obtained by Topology A and B, we see that in all non-trivial cases, Topology A outperforms Topology B.

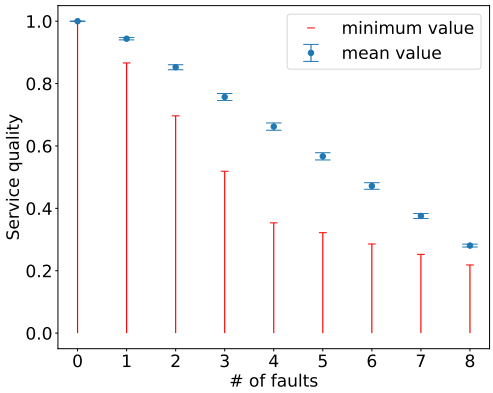


Figure 7. Average and minimal service quality together with standard deviation error bars for a WSN according to Topology B.

Topology C: This topology consists of three subsystems of identical nature composed in a parallel manner (see Figure 8)

A subsystem consists of a gateway node on a higher level

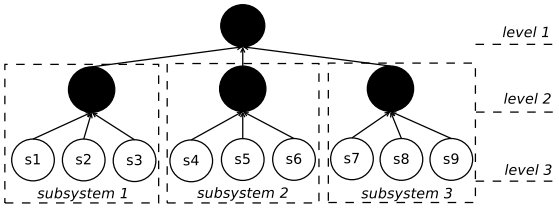


Figure 8. Topology C: WSN with three levels and $n = 9$ sensor nodes. Three subsystems of identical size are composed in a parallel fashion to form a new system.

and three sensor nodes on the next lower level. The three subsystems together with an additional gateway node, acting as root, form the 3-level topology. Note that within the three subsystems, sensor nodes are evenly distributed. Each subsystem is 2-region-adherent and the entire system is, again, 8-region-adherent. The test results are given in Table III and Figure 9.

Although Topology C differs from Topology A as Topology C composes three subsystems in a parallel manner (as opposed to composing nine sensor nodes in parallel), the values obtained for service quality, average service quality, standard deviation, and variance are very similar. The particular symmetric grouping of three sensor nodes in three subsystems has only a negligible effect on these measures.

TABLE III. TEST RESULTS OF THE WORST CASE, THE AVERAGE, THE STANDARD DEVIATION, AND THE VARIANCE OF THE SERVICE QUALITY OF A WSN AFTER EACH NUMBER OF FAULTS FOR TOPOLOGY C.

# of Faults	Minimum Service Quality	Average Service Quality	Standard deviation	Variance
0	1.0	1.0	0.0	0.0
1	0.9337	0.9474071917	$5.070 * 10^{-5}$	0.00712
2	0.8381	0.8505722057	$6.499 * 10^{-5}$	0.00806
3	0.7424	0.7573294496	$2.953 * 10^{-5}$	0.00543
4	0.6467	0.6617785970	$3.488 * 10^{-5}$	0.00590
5	0.5534	0.5664461319	$2.848 * 10^{-5}$	0.00533
6	0.4627	0.4718620317	$1.888 * 10^{-5}$	0.00434
7	0.3693	0.3764175436	$1.351 * 10^{-5}$	0.00367
8	0.2764	0.2813692449	$8.467 * 10^{-6}$	0.00290

Topology D: For this topology, we modify Topology C such that the sensor nodes are *not evenly* distributed to the three subsystems. We believe that this homogeneous distribution of sensor nodes to subsystems is responsible for the very low standard deviation and variance of Topology C. Topology D is given by Figure 10. As one can observe, the assignment of the nine sensor nodes to the three subsystems is now somehow "unbalanced." Subsystem 3 is 4-region-adherent, whereas subsystems 1 and 2 are only 1-region-adherent. The whole system is again 8-region-adherent. Table IV and Figure 11 show what we have expected. Topology D is responsible for a severe loss in minimal service quality compared to Topologies A and C. Furthermore, Topology D's standard deviation and variance is also much higher. When comparing the results of Topology D with the results of an hierarchical composition as given by Topology B, one observes that the former outperforms the latter for all non-trivial cases in terms of minimal service quality, standard deviation, and variance.

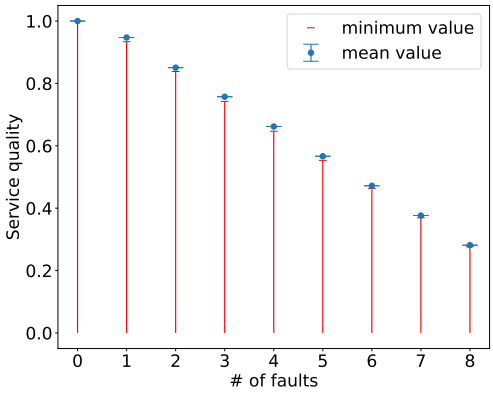


Figure 9. Average and minimal service quality together with standard deviation error bars for a WSN according to Topology C.

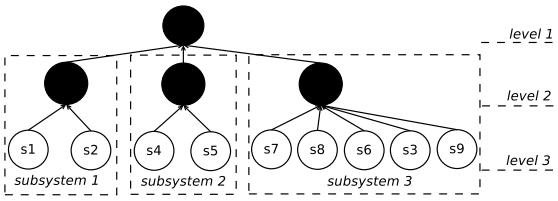


Figure 10. Topology D: WSN with three levels and $n = 9$ sensor nodes, where the sensor nodes are on the same level but not equally distributed among the three subsystems.

Topology E: In this topology, shown in Figure 12, we increase the number of levels by applying a hierarchical composition of systems twice. The resulting topology still has nine sensor nodes in total, but they are distributed on four levels. Within each level, though, the sensor nodes are evenly distributed. A WSN according to this topology can be interpreted as a system that uses a subsystem. This subsystem includes a subsystem (shown as "sub-subsystem" in the figure) of its own. The "parent" system spanning levels 1 and 2, as well as the subsystem and the sub-subsysten, are 2-region-adherent. The entire system is 8-region-adherent.

As we can observe in Table V and Figure 13, the standard deviation and the variance increase again. In particular, they are even higher than the corresponding values of Topology B. Furthermore, the minimal service quality of all non-trivial cases

TABLE IV. TEST RESULTS OF THE WORST CASE, THE AVERAGE, THE STANDARD DEVIATION, AND THE VARIANCE OF THE SERVICE QUALITY OF A WSN AFTER EACH NUMBER OF FAULTS FOR TOPOLOGY D.

# of Faults	Minimal Service Quality	Average Service Quality	Standard deviation	Variance
0	1.0	1.0	0.0	0.0
1	0.8881	0.9479234514	0.0023	0.04891
2	0.7518	0.8535155883	0.0033	0.05829
3	0.6037	0.7565725916	0.0041	0.06452
4	0.4698	0.6612940253	0.0045	0.06727
5	0.4110	0.5664977363	0.0046	0.06830
6	0.3553	0.4720827765	0.0041	0.06423
7	0.2963	0.3751367238	0.0032	0.05683
8	0.2420	0.2821965632	0.0018	0.04298

 ISBN: 978-1-61208-633-0

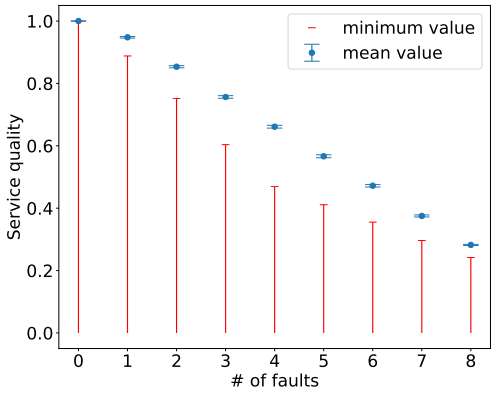


Figure 11. Average and minimal service quality together with standard deviation error bars for a WSN according to Topology D.

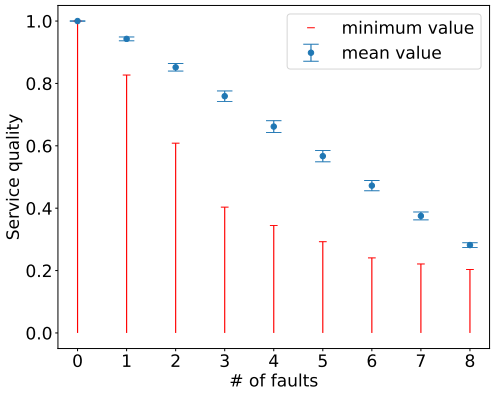


Figure 13. Average and minimal service quality together with standard deviation error bars for a WSN according to Topology E.

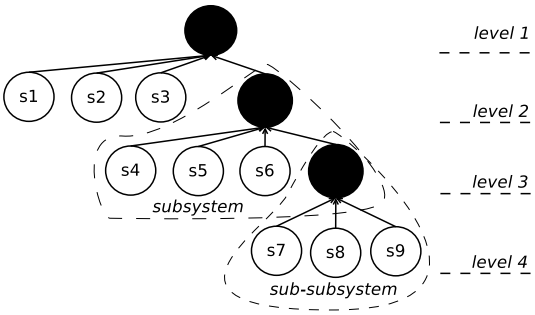


Figure 12. Topology E: WSN with four levels and $n = 9$ sensor nodes.

are the lowest among all topologies investigated. Through the hierarchical composition – introducing various levels where sensor nodes might be placed on – the overall impact of a failing sensor node on the observed measures depends on its level: a sensor node failing on a lower level (having a higher level number) has less an impact than a sensor node failing on a higher level.

V. Conclusion

In this paper, we presented a first step towards analyzing the impact of faults on the service quality of differently composed (i.e., structured) region-adherent WSNs. By means of parallel and hierarchical composition, we designed five tree-shaped network topologies. These topologies were subsequently used by WSNs to perform their tasks. Each WSN was – by design – 8-region-adherent. For each topology (and thereby: for each WSN), we calculated the minimal service quality. Furthermore, we simulated all possible fault scenarios and calculated the average, the standard deviation, and the variance of the service quality. The results showed that – depending on the topology – faults can differently impact the minimal service quality and related measures of the system. In particular, the experiments suggest that composing a WSN from subsystems in a parallel fashion leads to better results than composing it hierarchically. Furthermore, the more evenly the sensor nodes are distributed over the subsystems, the less negative is the impact of failures.

TABLE V. Test results of the worst case, the average, the standard deviation, and the variance of the service quality of a WSN after each number of faults for Topology E.

# of Faults	Minimal Service Quality	Average Service Quality	Standard deviation	Variance
0	1.0	1.0	0.0	0.0
1	0.8269	0.9428500325	0.0061	0.07837
2	0.6084	0.8517187577	0.0120	0.10988
3	0.4033	0.7591684317	0.0167	0.12945
4	0.3447	0.6617107143	0.0188	0.13727
5	0.2925	0.5669404963	0.0182	0.13499
6	0.2406	0.4723015525	0.0164	0.12812
7	0.2211	0.3751863283	0.0126	0.11241
8	0.2035	0.2817474513	0.0075	0.08663

Acknowledgment

This work has been funded by the German Research Foundation through the Research Training Group DFG-GRK 1765: “System Correctness under Adverse Conditions” (SCARE, scare.uni-oldenburg.de).

References

[1] P. Jalote, *Fault Tolerance in Distributed Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[2] C. P. Shelton, “Scalable Graceful Degradation of Distributed Embedded Systems,” Ph.D. dissertation, Carnegie Mellon University, Pittsburg, PA., U.S.A., 2003.

[3] J. S. Becker, D. Rahmatov, and O. Theel, “Dependable Systems through Region-Adherent Distributed Algorithms,” in *Proc. of the International Conference in Central Asia on Internet (ICI ’13)*. Tashkent, Uzbekistan: IEEE, Oct. 2013, pp. 203–212.

[4] ——, “Region-Adherent Algorithms: Restricting the Impact of Faults on Service Quality,” in *Proc. of the 20th IEEE Pacific Rim Intern. Symp. on Dependable Comp. (PRDC’14)*. Singapore: IEEE, November 2014, pp. 203–212.

[5] D. Rahmatov and O. Theel, “Towards a Design Theory of Region-Adherent Algorithms,” in *Proc. of the 23d Euromicro International Conference on Parallel, Distributed and Network-based Processing, Work-in-Progress (PDP 2015)*. Turku, Finland: IEEE, Mar. 2015, pp. 188–193.

[6] J. S. Becker, D. Rahmatov, and O. Theel, “Brief Announcement: Region-Adherent Algorithms – Bounding the Impact of Faults in Space,” in *Proc. of the 16th Intern. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS ’14)*. Paderborn, Germany: LNCS, Sep. 2014, pp. 362–365.

 ISBN: 978-1-61208-633-0

POMDP Based Throughput Maximization Scheme For Wireless Powered Cognitive Radio Network

Hiep Vu-Van and Insoo Koo
School of Electrical Engineering
University of Ulsan, Ulsan, South Korea
Email: vvhiep@gmail.com and iskoo@ulsan.ac.kr

***Abstract*—In wireless powered Cognitive Radio Network (CRN), a Cognitive radio User (CU) can receive the wireless power transferred from an energy source for its operation. In this paper, we consider the operations of a CU in the CRN with wireless powered transfer. The operations of the CU include *three* modes in terms of *Data Transmission* (DT), *Receive Wireless Power* (WP) and *Sleep* (SL). In order to consider the effect of the future operation on the current reward of the CU, we formulate the problem to choose an optimal operation mode of the CU as a Partially Observable Markov Decision Process (POMDP), in which the *two* states in terms of belief about the channel state and remaining energy are utilized as main factors to determine an optimal operation of the CU. Simulation results show the efficiency of the proposed scheme.**

***Index Terms*—cognitive radio; wireless powered transfer; operation mode switching; maximizing throughput; POMDP.**

I. Introduction

Cognitive Radio (CR) technology can improve spectrum utilization by allowing Cognitive radio Users (CU) to share the frequency assigned to a licensed user, called the Primary User (PU). In order to avoid interference with the operation of the licensed user, the CU is allowed to be active only when the frequency is free. Otherwise, when the presence of the PU is detected, the CU has to vacate their occupied frequency.

In the CRN, the CU often has a small battery that can maintain operations of the CU in a short time. The wireless power source [1][2] can charge the battery to extend its lifetime. However, the wireless power is still limited, therefore, the performance of the CRN strongly depends on how effectively the CU uses its power. The problem of optimal energy management has been considered previously in [3][4] where an optimal energy management scheme for a sensor node with an energy harvester to maximize throughput is proposed. In [5][6], a scheme to find an optimal action policy including sleeping to save energy or being active to take an opportunity of transmitting data is proposed. In [5], the CU has an independent energy harvester that harvests non-radio frequency (RF) energy from surround environment (e.g., wind, solar etc.), hence, the CU can perform data transmission and energy harvesting concurrently. On the other hand, [5] adopts an RF energy harvester that harvests energy from a PU's signal in the closed environment. Subsequently, the energy harvesting performance in [5][6] significantly fluctuates due to the uncontrollable wireless energy source.

In this paper, we consider the operation of wireless powered cognitive radio network that includes a control center, a wireless power source and cognitive radio user. The control center has a strong power supply (i.e., it always has enough energy to take any operation) and controls the operation of the whole network. Because the wireless power source can be controlled by the control center, the energy harvester of the CU is more efficient than conventional where the harvester receives energy from an unknown source. In the CRN, the operations of the CU includes *three* modes in terms of *Data Transmission* (DT), *Receive Wireless Power* (WP) and *Sleep* (SL). Due to the limitation of the hardware, the CU has a limited capacity battery that can be recharged by a wireless power source. On the other hands, the CU cannot perform multiples actions at the same time (i.e., it cannot perform data transmission and receiving energy at the same time).

This paper will propose a scheme to determine an optimal operation for the CU that can maximize the average achieved throughput of the CRN. In order to take a long-term optimization for the operation selection, we formulate the problem to choose an optimal operation as a partially observable Markov decision process (POMDP) in which the CU's *two* states in terms of the belief about the absence probability of the primary user (PU) and the remaining energy are utilized as main factors to decide an optimal operation of the CU in current time slot. By applying POMDP, the effects of future operations on the current achieved throughput are also considered. It is expected that the proposed scheme based on POMDP theory will provide the CR system an improved performance.

The remaining part of this paper is organized as follows. Section 2 shows the system model of the considered wireless powered cognitive radio network. Section 3 formulates an optimization problem based on POMDP in order to find the optimal operation mode for maximizing throughput of the CRN. Section 4 introduces simulation model and simulation results of the proposed and reference schemes. Finally, Section 5 concludes this paper.

II. System Model

A wireless powered Cognitive Radio Network (CRN) shown in figure 1, that includes a control center, a wireless energy source and a CU, works in a cognitive manner, where the CRN is allowed to opportunely utilize the channel that is assigned to a primary user (PU). The operation of the PU changes between *two* states of the Markov chain model, that is, Presence (P) and Absence (A) as shown in figure 2. The transition probability of the PU from state P to state A, state A to state P and from

 ISBN: 978-1-61208-633-0

state A or state P to itself are defined as P_{PA}, P_{AP}, P_{AA} and P_{PP}, respectively.

The control center of the CRN controls the operations of CU that include *Sleep* (SL), *Receive Wireless Power* (WP) and *Data Transmission* (DT) modes. In *Sleep* mode, the control center does not request the CU to do any action, so the CU is in the sleep state. *Receive Wireless Power* mode firstly requests the wireless energy source to transfer energy to the CU, after that the energy source will wirelessly transfer energy to the CU where the energy will be received and stored in a limited capacity battery of the CU to use for future operation. In the *Data Transmission* mode, the control center firstly performs spectrum sensing to determine the state of the PU. If the sensing result is *Presence*, the control center does not ask the CU to do any action. Otherwise, if the sensing result is *Absence*, the control center asks the CU to transmit data. If the transmission successes, the CRN can achieve a throughput as:

$$R = \frac{T - t_s}{T} C_0 \tag{1}$$

where T is the duration of a time slot, t_s is the sensing time and C_0 is the throughput of the communication link, which is defined as $C_0 = \log_2(1 + SNR_{CR})$, where SNR_{CR} is the signal to noise ratio (SNR) received in the control center.

In the *Receive Wireless Power* mode, the CU receives the energy from the energy source. Let define $E_h(t)$ as the number of units of energy that the CU receives at the time slot t. Due to the fluctuation of wireless power transfer, the amount of energy that the CU received each time slot is not stable. Hence, we assume that $E_h(t)$ takes value from a finite number ξ_h of energy units.

$$E_h(t) \in \Upsilon_h = \left\{ e_1^h, e_2^h, ..., e_{\xi_h}^h \right\}, \tag{2}$$

where $0 \le e_1^h < e_2^h < ... < e_{\xi_h}^h \le E_{ca}$, and E_{ca} (units of energy) is the capacity of the battery of the CU.

The probability mass function (PMF) of the harvested energy is given as follows:

$$P_{E_h}(k) = \Pr\left[E_h(t) = e_k^h\right], \; k = 1, 2, ..., \xi_h. \tag{3}$$

We assume that the harvested energy follows the stochastic process that are marked by Poisson process. Subsequently,

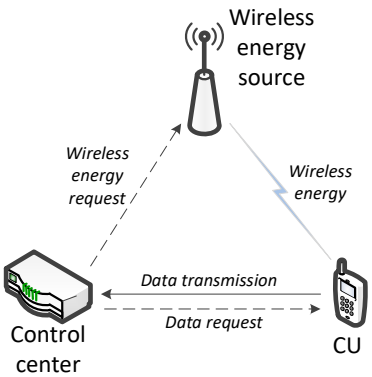


Figure 1. System model of the network.

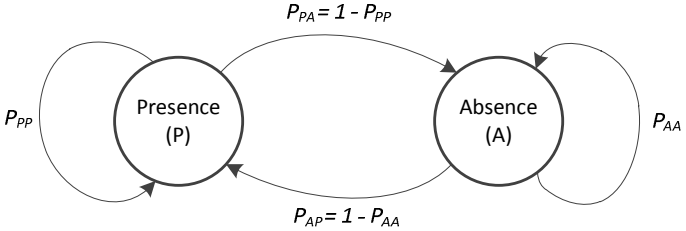


Figure 2. Markov chain states of the PU.

$E_h(t)$ is a Possion random variable with mean e_m^h and the PMF in (3) can be rewritten as follows:

$$P_{E_h}(k) \approx \frac{e^{-e_m^h} \left(e_m^h\right)^k}{k!}, \; k = 1, 2, ..., \xi_h. \tag{4}$$

III. Throughput Maximization Scheme Based on POMDP for Wireless Powered Cognitive Radio Network

In this section, in order to maximize its achieved throughput of the CRN, the problem of switching operation will be formulated as POMDP framework. As mentioned in section 2, we consider three operation modes of the CRN including *Sleep (SL)*, *Receive Wireless Power (WP)* and *Data Transmission(DT)* modes .

Hence, in order to formulate a POMDP, we define the operation space as: $\Psi = \{SL, \; WP, \; DT\}$. In addition, the state space of the system is defined as $S(t) = \{e_r(t), p_0(t)\}$, where $e_r(t)$ is the remaining energy of the CU and $p_0(t)$ is the belief about absence probability of the PU signal.

The reward of CRN is defines as its achieved throughput R according to the chosen operation. For POMDP framework, we define the value function $V(e_r(t), p_0(t))$ as the maximum total discounted throughput from the current time slot when the current state of the CU is $S(k)$. Subsequently, we have,

$$V(S(k)) = \max_{a(k)} \left\{ \sum_{t=k}^{\infty} \alpha^{t-k} R(S(t), a(t)) \, | S(k) \right\} \tag{5}$$

where $0 \le \alpha < 1$ is the discount factor, $S(k) = \{e_r(k), p_0(k)\}$. $R(S(t), a(t))$ is the throughput of the CU achieved at the t^{th} time slot, which is mainly dependent on state $S(t)$ and action decision $a(t)$.

A. Sleeping Mode (ϕ_1)

If the CU decides to remain sleeping, no throughput is achieved, then $R(S(t), SL \, | \phi_1) = 0$.

The belief p_0 for the next time slot will be updated as

$$p_0(t+1) = p_0(t) P_{AA} + (1 - p_0(t)) P_{PA}. \tag{6}$$

In practice, the CU may consume energy even when it is in *Sleep* mode. However, the consumed energy is very small. Therefore, we assume that the CU does not consume energy in the *Sleep* mode. Hence, the remaining energy of the CU keeps the same as previous time slot:

$$e_r(t+1) = e_r(t), \tag{7}$$

 ISBN: 978-1-61208-633-0

with transition probability

$$\Pr\left(e_r(t) \rightarrow e_r(t+1)\,|\phi_1\right) = 1. \quad (8)$$

B. Receive Wireless Power mode (ϕ_2)

In this operation mode, the CU does not transmit data. Therefore, no throughput is achieved, $R\left(S(t), WP\,|\phi_2\right) = 0$

This operation mode has no more information to update the belief, p_0, so that the belief p_0 is updated as the *Sleep mode* in (6).

In this operation mode, the CU will receive energy from the wireless power source. So that, the remaining energy of the CU will be increased as,

$$e_r(t+1) = \min\{e_r(t) + e_k^h\left(t\right), E_{ca}\},\ k = 1, 2, ..., \xi_h, \quad (9)$$

with transition probability,

$$\Pr\left(e_r(t) \rightarrow e_r(t+1)\,|\phi_1\right) = \Pr\left[E_h\left(t\right) = e_k^h\right]. \quad (10)$$

C. Data Transmission mode

This mode gives the CRN an opportunistic to achieve throughput by performing data transmission. So that, in order to transmit data the remaining energy of the CU must be higher than the energy spending for transmission, that is, $e_r(t) > e_t$.

The achieved throughput of the system depends on their observations. In this paper, we define *three* observations for the *Data Transmission* mode as follows:

Observation 1 *(ϕ_3):* The control center detects that the PU is *present* (the channel is used by the PU - state ***P***). Then, the CU is not asked to transmit data and there is no achieved throughput $R\left(S(t), DT\,|\phi_3\right) = 0$. The probability that this observation (ϕ_3) happens is:

$$\Pr\left(\phi_2\right) = p_0(t)P_f + (1 - p_0(t))P_d, \quad (11)$$

where P_f and P_d are spectrum sensing performance of the control center. P_f is the probability of false alarm that event happens when the PU signal is actually *absence*, however the sensing result is given as *presence*. P_d is the probability of detection that event happens when the sensing result is correct as the PU signal is *presence*.

The sensing result can be used to correct the belief p_0 in the current time slot as,

$$p_0^u(t) = \frac{p_0(t)P_f}{p_0(t)P_f + (1 - p_0(t))P_d}. \quad (12)$$

As a results, the updated belief for the next time slot is given by:

$$p_0(t+1) = p_0^u(t)P_{AA} + (1 - p_0^u(t))P_{PA}. \quad (13)$$

Since, this observation of the *Data Transmission* mode does not require the CU to do any action, the energy of the CU is not changed. So that, the remaining energy and transition probability are updated as the *Sleep* mode in (7) and (8).

TABLE I. SIMULATION PARAMETERS

Symbol	Description	Value
SNR_{CR}	SNR of the sensing channel	-10 dB
$\Pr(H_0)$	The absence probability of the PU	0.5
P_{AA}	Transition probability from state A to itself	0.8
P_{PA}	Transition probability from state P to state A	0.2
E_{ca}	Total capacity of battery	15 units of energy
e_m^h	mean of harvested energy	2 units of energy
e_t	Transmission energy	2 units of energy

Observation 2 *(ϕ_4):* The control center does not detect any signal from the PU (i.e., the state *A* of the PU). The CU is requested to transmit its data to the control center through the channel assigned to the PU and the transmission is success (i.e., the CU can receive an ACK message). This means that the sensing results is correct (the PU signal is really absent). The throughput is achieved as:

$$R\left(S(t), DT\,|\phi_4\right) = \frac{T - t_s}{T} C_0. \quad (14)$$

The probability that the observation ϕ_4 happens is:

$$\Pr\left(\phi_3\right) = p_0\left(t\right)\left(1 - P_f\right). \quad (15)$$

The belief and remaining energy for the next time slot can be updated, respectively, as:

$$p_0(t+1) = P_{AA} \quad (16)$$

and

$$e_r(t+1) = e_r(t) - e_t, \quad (17)$$

where e_t is the energy spent for data transmission.

The transition probability of the updated energy is given as,

$$\Pr\left(e_r(t) \rightarrow e_r(t+1)\,|\phi_4\right) = 1. \quad (18)$$

Observation 3 *(ϕ_5):* This observation is similar to the observation ϕ_4, state A of the PU is detected and the CU transmits its data. However, the CU can not receive ACK message. This means that the sensing results is incorrect (the PU signal is *present*), and the transmission data fails, no throughput is achieved, $R\left(S(t), DT\,|\phi_5\right) = 0$. The probability that ϕ_5 is obtained is:

$$\Pr\left(\phi_5\right) = \left(1 - p_0\left(t\right)\right)\left(1 - P_d\right). \quad (19)$$

The belief that the PU is in state A at the next time slot is:

$$p_0(t+1) = P_{PA} \quad (20)$$

The remaining energy and transition probability of the CU for the next time slot can be updated similar to the case of observation ϕ_4 in (17) and (18), respectively.

According to those observations, the expected value function in (5) can be expressed as (21). In order to find an optimal mode policy for maximizing throughput, the optimization problem in (21) will be solved by using the *value iterations* method [7].

 ISBN: 978-1-61208-633-0

$$V\left(S\left(k\right)\right)=\max_{a_k}\left\{\sum_{t=k}^{\infty}\alpha^{t-k}\sum_{\phi_i\in a(t)}\Pr\left(\phi_i\right)\sum_{e_r(t+1)}\Pr\left(e\left(t\right)\rightarrow e\left(t+1\right)|\phi_i\right)R\left(S\left(t\right),a\left(t\right)|\phi_i\right)|S\left(k\right)\right\} \quad (21)$$

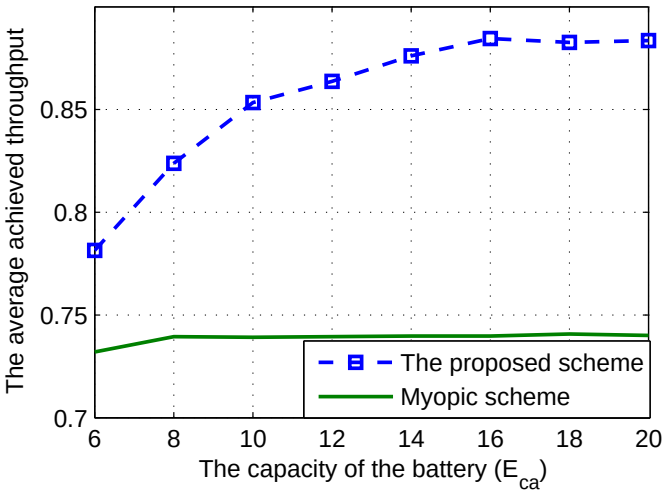


Figure 3. The average achieved throughput of the system versus battery capacity E_{ca} when $e_t = 2$ and $e_m^h = 1$.

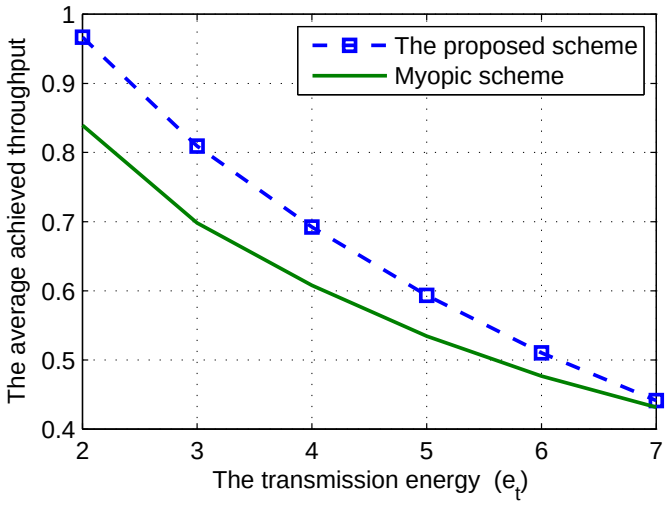


Figure 5. The average achieved throughput of the system versus transmission energy e_t when $e_m^h = 2$ and $E_{ca} = 15$.

IV. Simulation Results

In this section, we present simulation results to show the efficiency of the proposed scheme. *Myopic* scheme only considers the current time slot for the *value function* (i.e. $\alpha = 0$) to determine the operation mode. This means that *Myopic* scheme always chooses the third operation (i.e., *Data Transmission*) when the CU has enough energy to perform transmission (i.e., $e_r > e_t$). Otherwise, they will choose the second operation (i.e., *Receive Wireless Power*. Simulation results of *Myopic* will be provided in this section for a reference. The parameters for simulations are shown in Table I.

In order to evaluate the performance of the proposed scheme, the average achieved throughput in each time slot of the system will be utilized.

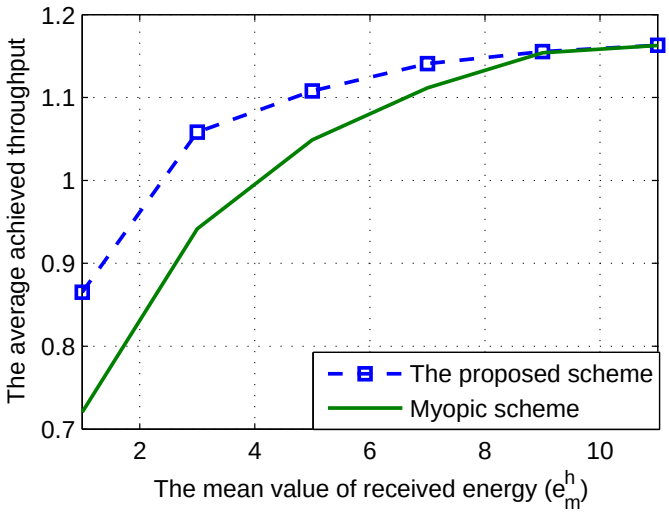


Figure 4. The average achieved throughput of the system versus the mean value of received energy e_m^h when $E_{ca} = 15$ and $e_t = 2$.

Figure 3 shows the average achieved throughput (bits/s/Hz) of the considered schemes according to the capacity of the CU battery E_{ca} (units of energy). The proposed scheme always achieves better performance in comparison with *Myopic* scheme as shown in the figure. It can be seen that the increase of battery capacity may help the CU to achieve higher throughput. However, when the battery is big enough (i.e., it has enough space to store all received energy), the increase of battery will not affect the throughput. However, since *Myopic* scheme always choose *Data Transmission* mode when it has enough energy so that the received energy is often spent before the battery is fully charged. Subsequently, the battery capacity seems not to affect on *Myopic scheme*.

Figure 4 presents the relation between the average achieved throughput and the mean value of the received energy e_m^h (units of energy) of the CU. The higher amount of e_m^h provides more energy for *Data Transmission* mode of the CU so that the CU can get more throughput. However, when e_m^h is high enough for *Data Transmission* mode of the CU in all time, the increase of e_m^h has no more effect on the CU's throughput. In this case, energy constrained will be disappeared, so that the chosen operation of the proposed scheme and *Myopic* scheme will be the same and that is the reason why they have the same performance when the received energy is high enough.

The effect of the transmission energy e_t (units of energy) on the average achieved throughput of the proposed scheme is shown in Figure 5. The higher energy spent for transmission may reduce the throughput of the system. However, the proposed scheme always obtains better performance in comparison with *Myopic scheme*.

 ISBN: 978-1-61208-633-0

Simulations results shown in all figures show that the proposed scheme can offer the CRN a better performance than the conventional *Myopic* scheme. That benefit of the proposed scheme is achieved by considering the future operations on the current achieved throughput.

V. Conclusion

In this paper, we consider the operations of wireless powered cognitive radio network in which the CU is powered by a wireless power source. The proposed scheme based on POMDP will determine an optimal operation of the CU to maximize the average achieved throughput with limited power supply. In the other hand, the proposed scheme can take consideration the effect of the future operation on the current throughput of the system by applying POMDP theory. Simulation results show that the proposed scheme can obtain better performance than conventional *Myopic* scheme.

Acknowledgement

This work was supported by the National Research Foundation (NRF) of Korea funded by the MEST (2017R1D1A1B03029448).

References

[1] Y. Wang, Y. Wang, F. Zhou, Y. Wu, and H. Zhou, "Resource allocation in wireless powered cognitive radio networks based on a practical non-linear energy harvesting model," *IEEE Access*, vol. 5, pp. 17 618–17 626, 2017.

[2] S. A. Mousavifar, Y. Liu, C. Leung, M. Elkashlan, and T. Q. Duong, "Wireless energy harvesting and spectrum sharing in cognitive radio," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014, pp. 1–5.

[3] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *IEEE Transactions on Wireless Communications,*, vol. 9, no. 4, pp. 1326–1336, 2010.

[4] S. Mao, M. H. Cheung, and V. Wong, "An optimal energy allocation algorithm for energy harvesting wireless sensor networks," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 265–270.

[5] A. Sultan, "Sensing and transmit energy optimization for an energy harvesting cognitive radio," *Wireless Communications Letters, IEEE*, vol. 1, no. 5, pp. 500–503, 2012.

[6] S. Park et al., "Optimal mode selection for cognitive radio sensor networks with rf energy harvesting," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, 2012, pp. 2155–2159.

[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientic, 2nd edition, 2001, vol. 1 and 2.

 ISBN: 978-1-61208-633-0

Scalable Messaging for Java-based Cloud Applications

Kevin Beineke, Stefan Nothaas and Michael Schöttner

Institut für Informatik, Heinrich-Heine-Universität Düsseldorf,
Universitätsstr. 1, 40225 Düsseldorf, Germany
E-Mail: Kevin.Beineke@uni-duesseldorf.de

***Abstract*—Many big data and large-scale cloud applications are written in Java or are built using Java-based frameworks. Typically, application instances are running in a data center on many virtual machines which requires scalable and efficient network communication. In this paper, we present the practical experience of designing a Java.nio transport for DXNet, a low latency and high throughput messaging system which goes beyond message passing by providing a fast parallel object serialization. The proposed design uses a zero-copy send and receive approach avoiding copying data between de-/serialization and send/receive. It is based on Java.nio socket channels complemented by application-level flow control to achieve low latency and high throughput for >10 GBit/s Ethernet networks. Furthermore, a scalable automatic connection management and a low-overhead interest handling provides an efficient network communication for dozens of servers, even for small messages (< 100 bytes) and all-to-all communication pattern. The evaluation with micro benchmarks shows the efficiency and scalability with up to 64 virtual machines in the Microsoft Azure cloud. DXNet and the Java.nio-based transport are open source and available on Github.**

Keywords–Message passing; Ethernet networks; Java; Data centers; Cloud computing;

I. INTRODUCTION

Big data processing is emerging in many application domains whereof many are developed in Java or are based on Java frameworks [1][2][3]. Typically, these big data applications aggregate the resources of many virtual machines in cloud data centers (on demand). For data exchange and coordination of application instances, an efficient network transport is very important. Fortunately, public cloud data centers already provide 10 GBit/s Ethernet and faster.

Java applications have different options for exchanging data between Java servers, ranging from high level Remote Method Invocation (RMI) [4] to low-level byte streams using Java sockets [5] or the Message Passing Interface (MPI) [6]. However, none of the mentioned possibilities offer high performance messaging, elastic automatic connection management, advanced multi-threaded message handling and object serialization all together. Therefore, we proposed DXNet [7], a network messaging system which meets all of these requirements. DXNet is extensible by transport implementations to support different network interconnects.

In this paper, we propose an Ethernet transport implementation for DXNet, called EthDXNet. The transport is based on Java.nio and provides high throughput and low latency networking over Ethernet connections.

The contributions of this paper are:

- the design of EthDXNet and practical experiences including:
  - scalable automatic connection management
  - zero-copy approach for sending and receiving data over socket channels
  - efficient interest handling
- evaluations with up to 64 VMs in the Microsoft Azure cloud

The evaluation shows that EthDXNet scales well while per-node message throughput and request-response latency is constant from 2 to 64 nodes, even in an high-load all-to-all scenario (worst case). Furthermore, high throughput is guaranteed for small 64-byte messages and saturation of the physical network bandwidth (5 GBit/s) with 4 KB messages. The latency experiments also show that EthDXNet efficiently utilizes the underlying network as long as the CPU does not get overstressed by too many application threads leading to a natural increase in latency.

The structure of the paper is as follows: after discussing related work, we present an overview of DXNet in Section III. In Section IV, we describe the sending and receiving procedure of EthDXNet, followed by a presentation of the connection management in Section V. Section VI focuses on the flow control implementation and Section VII on the interest handling. The evaluation is in Section VIII. The conclusions can be found in Section IX.

II. RELATED WORK

In this section, we discuss the related work for this paper.

A. JavaRMI

Java's RMI [4] provides a high level mechanism to transparently invoke methods of objects on a remote machine, similar to Remote Procedure Calls (RPC). Parameters are automatically de-/serialized and references result in a serialization of the object itself and all reachable objects (transitive closure), which can be costly. Missing classes can be loaded from remote servers during RMI calls, which is very flexible but introduces even more complexity and overhead. Additionally, the built-in serialization is known to be slow and not very

 ISBN: 978-1-61208-633-0

space efficient [8][9]. Furthermore, method calls are always blocking.

B. MPI

MPI is the state-of-the-art message passing standard for parallel high performance computing. It is available for Java applications by implementing the MPI standard in Java or wrapping a native library. However, MPI was designed for spawning jobs with finite runtime in a static environment. DXNet's and EthDXNet's main application domain are ongoing applications with dynamic node addition and removal (not limited to). The MPI standard defines the required functionality for adding and removing processes, but common MPI implementations are incomplete in this regard [10][11]. Furthermore, job shutdown and crash handling is still improvable [11].

C. Java.nio

The `java.io` and `java.net` libraries provide basic implementations for exchanging data via TCP/IP and UDP sockets over Input- and OutputStreams [12][5]. To create a TCP/IP connection between two servers, a new `Socket` is created and connection established to a remote IP and port. On the other end, a `ServerSocket` must be listening on given IP-port tuple creating a new Socket when accepting an incoming connection-creation request. The connection creation must be acknowledged from both sides and can be used to exchange byte arrays by reading/writing from/to the Socket hereafter. While this is sufficient for small applications with a few connections, this basic approach lacks several performance-critical optimizations [13] introduced with `Java.nio` [12][14]. (1) Instead of byte arrays, the read/write methods of Java.nio use `ByteBuffers`, which provide efficient conversion methods for all primitive data types. (2) ByteBuffers can be allocated outside of the Java heap allowing system-level I/O operations on the data without copying as the ByteBuffer is not subject to the garbage collection outside of the Java heap. Obviously, this relieves the garbage collector as well lowering the overhead with many buffers. (3) `SocketChannels` and `Selectors` enable asynchronous, non-blocking operations on stream-based sockets. With simple Java Sockets, user-level threads have to poll (a blocking operation) in order to read data from a Socket. Furthermore, when writing to a Socket the thread blocks until the write operation is finished, even if the Socket is not ready. With Java.nio, operation interests (like READ or WRITE) are registered on a Selector which selects operations when they are ready to be executed. This enables efficient handling of many connections with a single thread. The dedicated thread is required to call the `select` method of the Selector which is blocking if no socket channel is ready or returns with the number of executable operations. All available operations (e.g., sending/receiving data) can be executed by the dedicated thread, afterwards.

D. Java Fast Sockets

Java Fast Sockets (JFS) is an efficient Java communication middleware for high performance clusters [15]. It provides the widely used socket API for a broad range of target applications and is compatible with standard Java compilers and VMs. JFS avoids primitive data type array serialization (JFS does not include a serializer), reduces buffering and unnecessary copies in the protocol and provides shared memory communication with an optimized transport protocol for Ethernet. DXNet provides a highly concurrent serialization for complex Java objects and primitive data types which avoids copying/buffering as well. EthDXNet is an Ethernet transport implementation for DXNet.

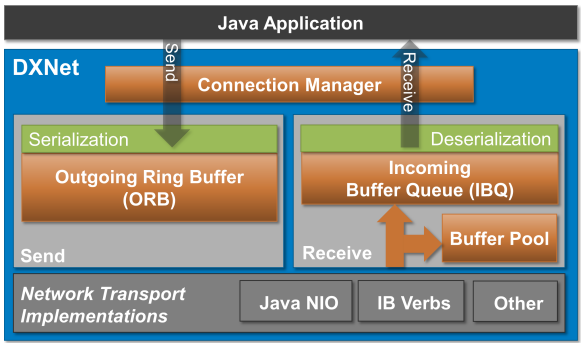


Figure 1. Simplified DXNet Architecture (from [7])

III. DXNET

DXNet is a network library for Java targeting, but not limited to, big data applications. DXNet implements an **event driven message passing** approach and provides a simple and easy to use application interface. It is optimized for highly multi-threaded sending and receiving of small messages by using **lock-free data structures, fast concurrent serialization, zero copy and zero allocation**. Split into two major parts, the core of DXNet provides automatic connection management, serialization of message objects and an interface for implementing different transports. Currently, an Ethernet transport using Java.nio sockets and an InifiniBand transport using ibverbs is implemented.

This section describes the most important aspects of DXNet and its core (see Figure 1) which are relevant for this paper. However, a more detailed insight of the core is given in a dedicated paper [7]. The source code is available at Github [16].

A. Connection Management

All nodes are addressed using an **abstract 16-bit node ID**. Address mappings must be registered to allow associating the node IDs of each remote node with a corresponding implementation dependent endpoint (e.g., socket, queue pair). To provide scalability with up to hundreds of simultaneous connections, our event driven system does not create one thread per connection. A **new connection is created automatically** once the first message is either sent to a destination or received from one. Connections are closed once a configurable connection limit is reached using a recently used strategy. Faulty connections (e.g., remote node not reachable anymore) are handled and cleaned up by the manager. Error handling on connection errors or timeouts are propagated to the application using exceptions.

B. Sending of Messages

Messages are Java objects and sent asynchronously. A message can be targeted towards one or multiple receivers. Using the message type **Request**, it is sent to one receiver. The

 ISBN: 978-1-61208-633-0

sender waits until receiving a corresponding response message (transparently handled by DXNet) or skips waiting and collects the response later.

One or multiple application threads call DXNet (concurrently) to send messages. Every message is automatically and concurrently serialized into the **Outgoing Ring Buffer (ORB)**, a natively allocated and lock-free ring buffer. When used concurrently, messages are automatically aggregated which increases send throughput. The ORB, one per connection, is allocated in native memory to allow direct and zero-copy access by the low level transport. The transport runs a decoupled dedicated thread which removes the serialized and ready to send data from the ORB and forwards it to the hardware.

C. Receiving of Messages

The network transport handles incoming data by writing it to **pooled native buffers**. We use native buffers and pooling to avoid burdening the Java garbage collection. Depending on how a transport writes and reads data, the buffer might contain fully serialized messages or just fragments. Every buffer is pushed to the ring buffer based **Incoming Buffer Queue (IBQ)**. Both, the buffer pool as well as the IBQ are shared among all connections. Dedicated **message handler threads** pull buffers from the IBQ and process them asynchronously by de-serializing them and creating Java message objects. The messages are passed to pre-registered callback methods of the application.

D. Flow Control

DXNet implements its own **flow control (FC)** mechanism to avoid flooding a remote node with messages. This would result in an increased overall latency and lower throughput if the remote node cannot keep up with processing incoming messages. When sending messages, the per connection dedicated FC checks if a configurable threshold is exceeded. This threshold describes the **number of bytes sent by the current node but not fully processed by the receiving node**. The receiving node counts the number of bytes received and sends a confirmation back to the source node in regular intervals. Once the sender receives this confirmation, the number of bytes sent but not processed is reduced. If an application send thread was previously blocked due to exceeding this threshold, it can now continue with processing the message.

E. Transport Interface

DXNet provides a transport interface allowing implementations of different transport types. One of the implemented transports can be selected on the start of DXNet. The transport and its specific semantics are transparent to the applications.

The following methods must be implemented for every transport:

- Setup connection
- Close and cleanup connection
- Signal to send data available in the ORB of a connection (callback)
- Pull data from the ORB and send it
- Push received raw data/buffer to the IBQ

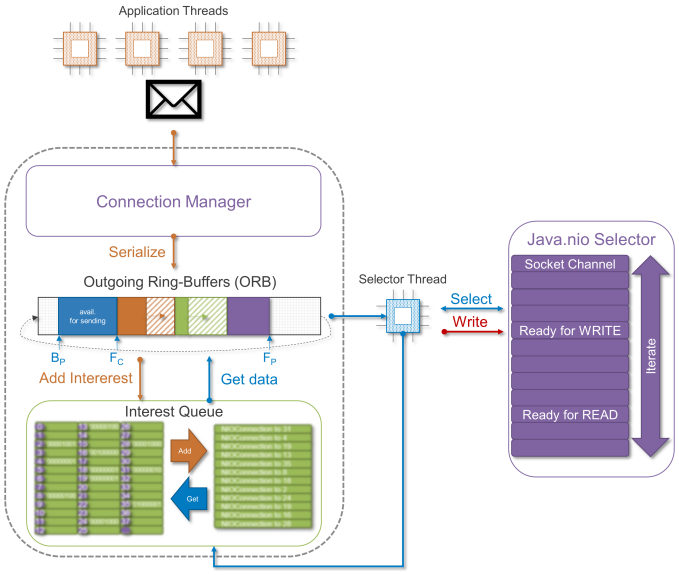


Figure 2. Data structures and Threads. Details of the Interest Queue can be found in Figure 4.

IV. EthDXNet - Sending and Receiving

In the following sections, we describe the Ethernet transport of DXNet, called EthDXNet. An overview of the most important data structures and threads of EthDXNet are depict in Figure 2.

A. Sending of Data

To send messages, the DXNet API methods `sendMessage` or `sendSync` are called by the application threads (or message handler threads). In DXNet, messages are always sent asynchronously, i.e., application threads might return before the message is transfered. It is possible, though, to wait for a response before returning to the application (sendSync). After getting the `ConnectionObject` (a Java object) from the **Connection Manager**, the message is serialized into the ORB associated with the connection. For performance reasons, many application threads can serialize into the same or different ORBs in parallel (more in [7]). The actual message transfer is executed by the **SelectorThread**, a dedicated daemon thread driving the Java.nio back-end. Thus, after serializing the message into the ORB, the application thread must signal data availability for the corresponding connection. This is done by registering a `WRITE` interest (see Table I) for given connection in the **Interest Queue** (see Section VII). When ready, Java.nio's **Selector** wakes-up the SelectorThread (which is blocked in the `select` method of the Selector) to execute the operation and thus transferring the message.

After returning from the select method, a **SelectionKey** is available in the ready-set of the Selector. It contains the operation interest WRITE, the socket channel and attachment (the associated ConnectionObject). This SelectionKey is dispatched based on the operation. In order to send the message over the network, the SelectorThread pulls the **data block** from the ORB of the corresponding connection and calls the `write` method of the socket channel. From this point, we cannot distinguish single messages anymore because

 ISBN: 978-1-61208-633-0

messages are naturally aggregated to data blocks in the ORBs, which is a performance critical aspect. The write method is called repeatedly until all bytes have been transferred or the method returned with return value 0. The second case indicates congestion on the network or the receiver and is best handled by stopping the transfer and continue it later. After sending, the back position (B_p, see Figure 2) of the ORB is moved by the number of bytes transferred to free space for new messages to send. Additionally, if the transfer was successful and the ORB is empty afterwards, the SelectionKey's operation is set to `READ` which is the preset operation and enables receiving incoming data blocks. If the transfer failed, the connection is closed (see Section V). If the transfer was incomplete or new data is available in the ORB, the SelectionKey is set to `READ | WRITE` (combination of READ and WRITE by using the bitwise or-operator) which triggers a new WRITE operation when calling select the next time but also allows receiving incoming messages. It is important to change the SelectionKey to this state as keeping only the WRITE operation could result in a deadlock situation in which both ends try to transfer data but none of them are able to receive data on the same connection. This causes the **kernel socket receive buffers** to fill up on both sides preventing further data transfer.

The ORB is a ring buffer allocated in native memory (outside of the Java heap). In order to pass a ByteBuffer to the socket channel, which is required for data transfer, we wrap a **DirectByteBuffer** onto the ORB and set the ByteBuffer's position to the front position in the ORB and the limit to the back position. A DirectByteBuffer is a ByteBuffer whose underlying byte array is stored in native memory and is not subject to garbage collection. This enables native operations of the operating system without copying the data first. The socket channel's send and receive operations are examples for those native operations, thus, benefiting from the DirectByteBuffer. Java does not support changing the address of a ByteBuffer. Therefore, on initialization of the ORB, we allocate a new DirectByteBuffer by calling `allocateDirect` of the Java object ByteBuffer and use the underlying byte array as the ORB. To do so, we need to determine the memory address of the byte array, which can be obtained with `Buffer.class.getDeclaredField("address")`. That is, during serialization the ORB is accessed with `Java.unsafe` by reading/writing from/to the actual address outside of the Java heap, but the socket channel accesses the data by using the DirectByteBuffer's reference (with adjusted position and limit). We do not access the ORB by using the DirectByteBuffer during serialization because of performance and compatibility reasons described in [7].

Although this approach prevents copying the data to be sent on user-level, the data is still copied from the ORB to the **kernel socket send buffer** which is a necessity of the stream-based socket approach. Therefore, configuring the kernel socket buffer sizes (one for sending and one for receiving) correctly has a great impact on performance. We empirically determined setting both buffer sizes to the ORBs' size offers a good performance without increasing the memory consumption too much (typically the ORBs are between 1 and 4 MB depending on the application use case).

B. Receiving of Data

Receiving messages is always initiated by Java.nio's **Selector** which detects incoming data availability on socket channels. When a socket channel is ready to be read from, the SelectorThread selects the SelectionKey and dispatches the READ operation. Next, the SelectorThread reads repeatedly by calling the `read` operation on the socket channel until there is nothing more to read or the buffer is full. If reading from the socket channel failed, the socket channel is closed. Otherwise, the ByteBuffer with the received data is flipped (limit = position, position = 0) and pushed to the IBQ (see Figure 1). The buffer processing is explained in [7].

In order to read from the socket channel, a ByteBuffer is required to write the incoming data into. Constantly allocating new ByteBuffers decreases the performance drastically. Therefore, we implemented a **buffer pool**. The buffer pool provides ByteBuffers, allocated in native memory (which are DirectByteBuffers), in different configurable sizes (e.g., 8×256 KB, 256×128 KB and 4096×16 KB). The SelectorThread pulls DirectByteBuffers using a worst-fit strategy as the amount of bytes ready to be received on the stream is unknown. It can also scale-up dynamically, if necessary. The buffer pool management consists of three lock-free ring buffers optimized for access of one consumer and N producers [7].

The pooled DirectByteBuffers are wrapped to provide the ByteBuffer's reference as well as the ByteBuffer's address. The reference is used for reading from the socket channel and the address is necessary to deserialize the messages within the ByteBuffer.

V. Automatic Connection Management

For sending and receiving messages, we have to manage all open connections and create/close connections on demand. A connection is represented by an object (ConnectionObject), containing a node ID to identify the connection based on the destination, a **PipeIn** and a **PipeOut**. The PipeOut consists mostly of an ORB, a socket channel and flow control for outgoing data. The PipeIn contains a socket channel, flow control for incoming data, has access to the buffer pool (shared among all connections) and more data structures important to buffer processing, which are not further discussed in this paper.

1) Connection Establishment: Connections are created in two ways: (1) actively by creating a new connection to a remote node or (2) passively by accepting a remote node's connection request. In both cases, the connection manager must be updated to administrate the new connection. Figure 3 shows the procedure of creating a new connection (active on the left side and passive on the right). The core part is the TCP handshake, which can be seen in the middle.

Active connection creation: A connection is created actively, if an application thread wants to send a message to a not yet connected node. To establish the connection, the application thread creates a new ConnectionObject (including PipeIn and PipeOut and all its components), opens a new socket channel and connects the socket channel to the remote node's IP and port. Afterwards, the application thread registers a `CONNECT` operation, creates a `ReentrantLock` and `Condition` and waits until the Condition is signaled or

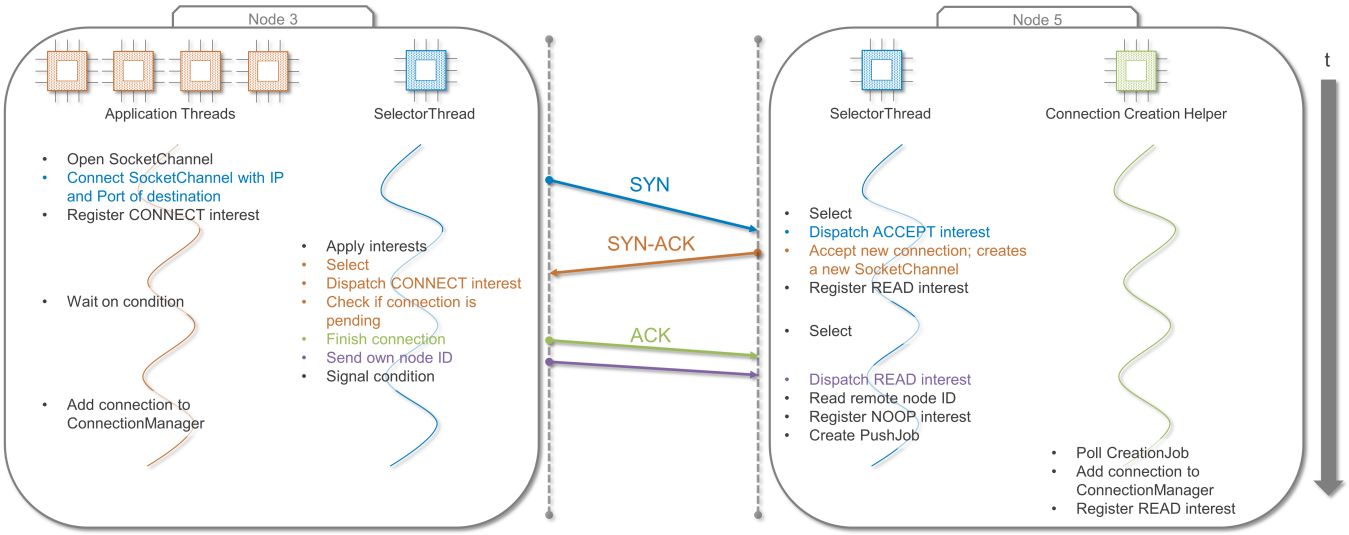


Figure 3. Connection Creation

the connection creation was aborted. To correctly identify the corresponding ConnectionObject to a socket channel, the ConnectionObject is attached to the SelectionKey when registering the CONNECT interest and all following interests.

The SelectorThread continues the connection establishment by applying the CONNECT interest and selecting the socket channel when the remote node accepted the connection or the connection establishment failed. After selecting the SelectionKey, the socket channel's status is checked. If it is pending, the connection creation was successful so far and the socket channel can be completed by calling `finishConnect`. If the connection establishment was aborted, the application thread is informed by setting a flag (which is checked periodically by the application thread).

The remote node has to identify the new node currently creating a connection. Thus, the node ID is sent to the remote node on the newly created channel. Furthermore, the SelectorThread marks the PipeOut as connected and signals the condition so the application thread can continue. The application thread adds the connection to the connection manager, increments the connection counter and starts sending data, afterwards.

Passive connection creation: For accepting and creating an incoming connection, the Selector implicitly selects a SelectionKey with `ACCEPT` operation interest which is processed by the SelectorThread by calling `accept` on the socket channel. This creates a new socket channel and acknowledges the connection. Afterwards, the interest `READ` is registered in order to receive the node ID of the remote node. After selecting and dispatching the interest, the node ID is read by using the socket channel's read method.

At this point the socket channel is ready for sending and receiving data, but the ConnectionObject has yet to be created and pushed to the connection manager. This process is rather time consuming and might be blocking if an application thread creates a connection to the same node at the same time (connection duplication is discussed in Section V-2). Therefore, the SelectorThread creates a job for creating the connection and forwards it to the **ConnectionCreationHelper** thread. Additionally, the interest is set to `NO-OP` (0) to avoid receiving data before the connection setup is finished and the connection is attached to the SelectionKey.

The ConnectionCreationHelper polls the job queue periodically. There are two types of jobs: (1) a connection creation job and (2) a connection shutdown job. The latter is explained in Section V-3. When pulling a connection creation job, the ConnectionCreationHelper creates a new ConnectionObject (including the pipes, ORB, FC, etc.) and registers a READ interest with the new ConnectionObject attached. Furthermore, the PipeIn is marked as connected.

To be able to accept incoming connection requests, every node must open a `ServerSocketChannel`, bind it to a well-known port and register the `ACCEPT` interest. Furthermore, for selecting socket channels, a `Selector` has to be created and opened.

2) Connection Duplication: It is crucial to avoid connection duplication which occurs if two nodes create a connection to each other simultaneously. In this case, the nodes might use different connections to send and receive data which corrupts the message ordering and flow control. There are two approaches for resolving this problem: (1) detecting connection duplication during/after the connection establishment and (2) avoiding connection duplication by using two separate socket channels for sending and receiving.

Solution 1: **Detect and resolve connection duplication** by keeping one connection opened and closing the other one. Obviously, the other node must decide consistently which can be done by including the node IDs (e.g., always keep the connection created by the node with higher node ID). One downside of this approach is the complex connection shutdown. It must ensure that all data initially to be sent over the closing connection has been sent and received. Furthermore, message ordering cannot be guaranteed until the connection duplication situation is resolved.

Solution 2: **Avoid connection duplication** by using two socket channels per connection: one for sending and one for receiving (implemented in EthDXNet). Thus, simultaneous connection creation leads to **one** ConnectionObject with

 ISBN: 978-1-61208-633-0

opened PipeIn and PipeOut (one socket channel, each) whereas a single connection creation opens either the PipeOut (active) or PipeIn (passive). This approach requires additional memory for the second socket channel, Java.nio's Selector has more socket channels to manage and connection setup is required from both ends. The additional memory required for the second socket channel is negligible as the kernel socket buffers are configured to use a very small socket receive buffer for the outgoing socket channel and a very small socket send buffer for the incoming socket channel. The second TCP handshake (for connection creation, both sides need to open and connect a socket channel) is also not a problem as both socket channels can be created simultaneously and for a long running big data application connections among application instances are typically kept over the entire runtime. Finally, the overhead for Java.nio's Selector is difficult to measure but is certainly not the bottleneck taking into account the limitations of the underlying network latency and throughput. **Sending out-of-band (OOB) data** is possible by utilizing **the unused back-channel of every socket channel**. We use this for sending flow control data in EthDXNet (see Section VI).

3) Connection Shutdown: Connections are closed on three occasions: (1) if a write or read access to a socket channel failed, (2) if a new connection is to be created but the configurable connection limit is reached or (3) on node shutdown. In the first case, the SelectorThread directly shuts down the connection. In the second case, the application thread registers a `CLOSE` interest to let the SelectorThread close the connection asynchronously. On application shutdown, all connections are closed by one **Shutdown Hook** thread.

To shut down a connection, first, the outgoing and incoming socket channels are removed from the Selector by canceling the SelectionKeys representing a socket channel's registration. Then, the socket channels are closed by calling the socket channels' `close` method. At last, the connection is removed from the connection manager by creating a shutdown job handled by the ConnectionCreationHelper (case (1)) or directly removing it when returning to the connection management (cases (2) and (3)). The ConnectionCreationHelper also triggers a `ConnectionLostEvent`, which is dispatched to the application for further handling (e.g., node recovery).

When dismissing a connection (case (2)), directly shutting down a connection might lead to data loss. Therefore, the connection is closed gracefully by waiting for all outstanding data (in the connection's ORB) to be sent. Priorly, the connection is removed from connection management to prevent further filling of the ORB. Afterwards, a CLOSE interest is registered to close the socket channels asynchronously. The SelectorThread does not shut down the socket channels on first opportunity but postpones shutdown for at least two RTT timeouts to ensure all responses are received for still outstanding requests.

VI. Flow Control

DXNet provides a flow control on application layer to avoid overwhelming slower receivers (see Section III). EthDXNet uses the *Transmission Control Protocol* (TCP) which already implements a flow control mechanism on protocol layer. Still, DXNet's flow control is beneficial when using TCP. If the application on the receiver cannot read and process the data fast enough, the sender's TCP flow control window, the maximum amount of data to be sent before data receipt has to be acknowledged by the receiver, is reduced. The decision is based on the utilization of the corresponding kernel socket receive buffer. In DXNet, reading incoming data from kernel socket receive buffers is decoupled from processing the included messages, i.e., many incoming buffers could be stored in the IBQ to be processed by another thread. Thus, the kernel socket receive buffers' utilizations do not necessarily indicate the load on the receiver leading to delayed or imprecise decisions by TCP's flow control.

This section focuses on the implementation of the flow control in EthDXNet. Flow control data has to be sent with high priority to avoid unintentional slow-downs and fluctuations regarding throughout and latency. Sending flow control data in-band, i.e., with a special message appended to the data stream, is not an option because the delay would be too high. TCP offers the possibility to send **urgent data**, which is a single byte inlined in the data stream and sent as soon as possible. Furthermore, urgent data is always sent, even if the kernel socket receive buffer on the receiver is full. To distinguish urgent data from the current stream (urgent data can be at any position within a message as transfer is not message-aligned), a dedicated flag within the TCP header needs to be checked. This flag indicates if the first byte of the packet is urgent data. Unfortunately, Java.nio does not provide methods for handling incoming TCP urgent data.

We solve this problem **by using both unused back-channels** of every socket channel which are available because of the double-channel connection approach in EthDXNet. Thus, the incoming stream of the outgoing socket channel and the outgoing stream of the incoming socket channel of every connection are used **for sending/receiving flow control data**.

Sending flow control data: When receiving messages, a counter is incremented by the number of received bytes for every incoming buffer. If the counter exceeds a configurable threshold (e.g., 60% of the flow control window), a `WRITE_FC` interest is registered. This interest is applied, selected and dispatched like any other WRITE interest. But, instead of using the socket channel of the PipeOut, **the PipeIn is used to send the flow control data**. The flow control data consists of one byte containing the number of reached thresholds (typically 1). If the threshold is smaller than 50%, for example 30%, it is possible that between registering the WRITE_FC interest and actually sending the flow control data, the threshold has been exceeded again. For example, if the current counter is 70% of the windows size which is more than two thresholds of 30%. In this case 2 * 30% = 60% is confirmed by sending the value 2. After sending flow control data, the SelectionKey is reset to READ to enable receiving messages on this socket channel, again.

Receiving flow control data: To be able to receive flow control data, the socket channel of **the PipeOut must be readable** (register READ). If flow control data is available to be received, the socket channel is selected by the Selector and the SelectorThread reads the single byte from the socket channel of the PipeOut. When processing serialized messages on the sender, a counter is incremented. Application threads which want to send further messages if the counter reached

 ISBN: 978-1-61208-633-0

TABLE I. JAVA.NIO INTERESTS

Interest	Description
OP_READ	channel is ready to read incoming data
OP_WRITE	set if data is available to be sent
OP_CONNECT	set to open connection
OP_ACCEPT	a connect request arrived

TABLE II. ETHDXNET INTERESTS

Interest	Description (refers to attached connection)
CONNECT	set OP_CONNECT for outgoing channel
READ_FC	set OP_READ for outgoing channel
READ	set OP_READ for incoming channel
WRITE_FC	set OP_WRITE for incoming channel
WRITE	set OP_WRITE for outgoing channel
CLOSE	shutdown both socket channels

the limit (i.e., the flow control window is full) are blocked until message receipt is acknowledged by the receiver. The read flow control value is used to decremented the counter to re-enable sending messages. Usually, the limit is never reached as the flow control data is received before (if the threshold on the receiver is low enough).

In Section IV-A, we discussed the end-to-end situation of both nodes sending data to each other, but never reading (if the SelectionKey's operation stays at WRITE) causing a deadlock. This situation cannot occur with two socket channels per connection as reading and writing are handled independently. But, a similar situation is possible where two nodes send data to each other, but flow control data is not read for a while. This does not cause a deadlock but decreases performance. By setting the interest to READ | WRITE, flow control data is read from time to time ensuring a contiguous high throughput.

VII. EFFICIENT MANAGEMENT OF OPERATION INTERESTS

Operation interests are an important concept in Java.nio and are registered in the Selector to create and accept a new connection, to write data or to enable receiving data. The operation interests are complemented by the ConnectionObject (as an attachment) and the socket channel stored together in a SelectionKey. As soon as the socket channel is ready for any registered operation, the Selector adds the corresponding SelectionKey to a ready-set and wakes-up the SelectorThread waiting in the `select` method. If the SelectorThread is not waiting in the select method, the next select call will return immediately. The SelectorThread can then process all SelectionKeys.

A. Types of Operations Interests

The operation interests can be classified into two categories: **explicit operation interests and implicit operation interests**. Implicit operations are registered as presets after socket channel creation and after executing explicit operations. For example, a READ interest is registered for a socket channel if data is expected to arrive on this socket channel. The operation is then selected implicitly by the Selector whenever data is available to be received. Another example is the ServerSocketChannel which implicitly accepts new incoming connection requests if the ACCEPT interest has been registered before. Explicit operations are single operations which need to be triggered explicitly by the application. For example, when the application wants to send a message, the application thread has to register a WRITE interest. When the socket channel is ready, the data is sent and the socket channel is set to the preset (in our case READ). It is not forbidden by Java.nio to keep explicit operations registered. But, as a consequence the operations are always selected (every time select is called) which increases CPU load and latency. Therefore, in EthDXNet, every explicit operation is finished by registering an implicit operation.

The set of Java.nio operation interests is extended by EthDXNet to support flow control and to enable closing connections asynchronously. Table I shows all interests specified by Java.nio and Table II lists all interests used in EthDXNet. The interests READ, WRITE and CONNECT are directly mapped onto OP_READ, OP_WRITE and OP_CONNECT. OP_ACCEPT is registered and selected by the Selector and must not be registered explicitly. READ_FC and WRITE_FC are used to register OP_READ and OP_WRITE interests for the back-channel used by the flow control. The interest CLOSE does not have a counterpart because the method `close` can be called explicitly on the socket channel.

B. Interest Queue

None of the interests in Table II are registered directly to the Selector because only the SelectorThread is allowed to add and modify SelectionKeys. This is enforced by the Java.nio implementation which blocks all register calls when the SelectorThread is waiting in the `select` method. This obstructs the typical asynchronous application flow and can even result in a deadlock if the Selector does not have implicit operations to select. This problem can be avoided by always waking-up the SelectorThread before registering the operation interest and synchronizing the register and select calls. However, this workaround results in a rather high overhead and a complicated work flow. Instead, **we address this problem with an Interest Queue** (see Figure 4) and register all interests in one bulk operation executed by the SelectorThread before calling select. This approach provides several benefits while solving the above problem: first, **the application threads can return quickly** after putting the operation interest into the queue and even faster (without any locking) when the interest was already registered (which is likely under high load). Second, **the operation interests can be combined** and put in a semantic order (e.g., CONNECT before WRITE) before registering (a rather expensive method call). Finally, **the operation interest-set can be easily extended**, e.g., by a CLOSE operation interest to asynchronously shut down socket channels.

Figure 4 shows the Interest Queue consisting of a byte array storing the operation interests of all connections (left side in Figure 4) and an ArrayList of ConnectionObjects containing connections with new operation interests sorted by time of occurrence (right side in Figure 4).

The byte array has one entry per node ID allowing access time in O(1). The node ID range is limited to 2^{16} (allowing

 ISBN: 978-1-61208-633-0

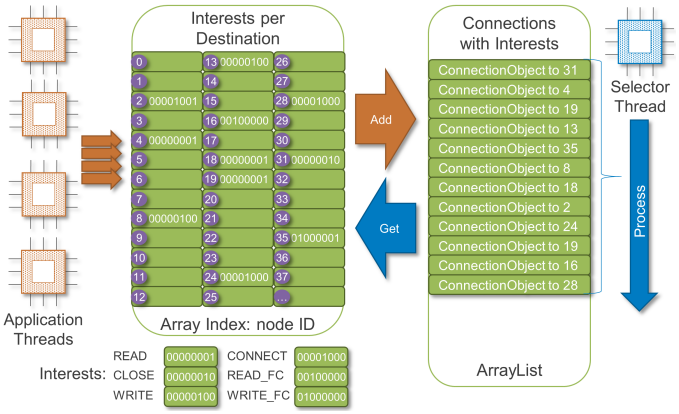


Figure 4. Interest Queue: the application threads add new interests to the Interest Queue. If interest was 0 before, the ConnectionObject is added to an ArrayList.

max. 65,536 nodes per application) which results in a fixed size of 64 KB for the byte array. An array entry is not zero if at least one operation interest was added for given connection to the associated node ID. Operation interests are combined with the bitwise or-operator to avoid overwriting any interest. By combining operation interests, the ordering of the interests for a single connection is lost. But, this is not a problem because a semantic ordering can be applied when processing them.

The ordering within the interests of one connection can be reconstructed but not the ordering across different connections. Therefore, whenever an interest is added to a non-zero entry of the byte array, the corresponding ConnectionObject is appended to an ArrayList. The order of operation interests is then ensured by processing the interest entries in the ArrayList in ascending order. The ArrayList also allows the SelectorThread to iterate only relevant entries and not all 2^{16}.

Processing operation interests: The processing is initiated either by the Selector implicitly waking up the SelectorThread if data is available to be read or an application thread explicitly waking up the SelectorThread if data is available to be sent. As waking-up the SelectorThread is a rather expensive operation (a synchronized native method call), it is important to call it if absolutely necessary, only. Therefore, the SelectorThread is woken-up after adding the first operation interest to the Interest Queue across all connections (the ArrayList is empty after processing the operation interests). If the SelectorThread is currently blocked in the select call, it returns immediately and can process the pending operation interests.

Listing 5 shows the basic processing flow of the SelectorThread. The first step in every iteration is to register all operation interests collected in the ArrayList of the Interest Queue. The SelectorThread gets the destination node ID from the ConnectionObject and the interests from the byte array. Operation interests are registered to the Selector in the following order:

1) CONNECT: register SelectionKey OP_CONNECT with given connection attached to an outgoing channel.
2) READ_FC: register SelectionKey OP_READ with given connection attached to an outgoing channel.
3) READ: register SelectionKey OP_READ with given connection attached to an incoming channel.

```
1  while (!closed) {
2    processInterests();
3
4    if (Selector.select() > 0) {
5      for (SelectionKey key :
           Selector.selectedKeys()) {
6        // Dispatch key
7        if (key.isValid()) {
8          if (key.isAcceptable()) {
9            accept();
10         } else if (key.isConnectable()) {
11           connect();
12         } else if (key.isReadable()) {
13           read();
14         } else if (key.isWritable()) {
15           write();
16         }
17       }
18     }
19   }
20 }
```

Figure 5. Workflow of SelectorThread

4) WRITE_FC: change SelectionKey of an incoming channel to OP_WRITE if it is not OP_READ | OP_WRITE.
5) WRITE: change SelectionKey of an outgoing channel to OP_WRITE if it is not OP_READ | OP_WRITE.
6) CLOSE: keep interest in queue for delay or close connection (see Section V-3).

The order is based on following rules: (1) a connection must be connected before sending/receiving data, (2) setting the preset READ is done after connection creation, only, (3) all READ and WRITE accesses must be finished before shutting down the connection and (4) the flow control operations have a higher priority than normal READ and WRITE operations. Furthermore, re-opening a connection cannot be done before the connection is closed and closing a connection is only possible if the connection has been connected before. Therefore it is not possible to register CONNECT and CLOSE together.

Finally, the processing of registered operation interests includes reseting the operation interest in the byte array and removing the ConnectionObject from the ArrayList.

VIII. Evaluation

We evaluated EthDXNet using up to 65 virtual machines (64 running the benchmark and one for deployment) connected with 5 GBit/s Ethernet in Microsoft's Azure cloud in Germany Central. The virtual machines are Standard_DS13_v2 which are memory optimized servers with 8 cores (Intel Xeon E5-2673), 56 GB RAM and a 10 GBit/s Ethernet connectivity, which is limited by SLAs to 5 GBit/s. In order to manage the servers, we created two identical scale-sets (as one scale-set is limited to 40 VMs) based on a custom Ubuntu 14.04 image with 4.4.0-59 kernel and Java 8.

We use a set of micro benchmarks for the evaluation sending messages or requests of variable size with a configurable number of application threads. All throughput measurements refer to the payload size which is considerably smaller than the full message size, e.g., a 64-byte payload results in 115 bytes to

 ISBN: 978-1-61208-633-0

TABLE III. ADDITIONAL PARAMETERS

Parameter	Value
ORB Size	4 MB
Flow Control Windows Size	2 MB
Flow Control Threshold	0.6
net.core.rmem_max	4 MB
net.core.wmem_max	4 MB

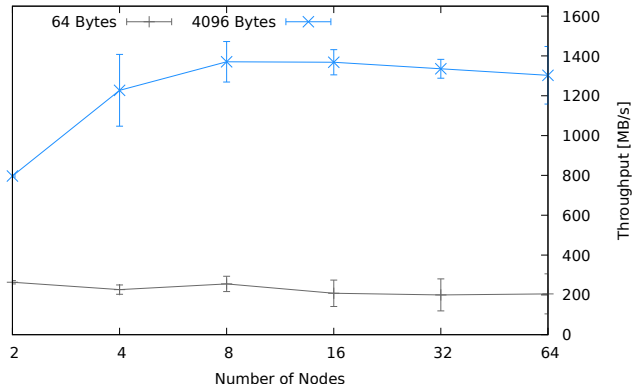


Figure 6. Message Payload Throughput per Node. 1 Application Thread, 2 Message Handler Threads

be sent on IP layer. Additionally, all runs with DXNet's benchmark are **full-duplex** showing the aggregated performance for concurrently sending and receiving messages/requests.

A. Message Throughput

First, we measured the asynchronous message throughput with an increasing number of nodes in an all-to-all test with message payloads of 64 and 4096 bytes. For instance, when running the benchmark with 32 nodes each node sends 25,000,000 64-byte messages to all 31 other nodes and therefore each node has to send and receive 775,000,000 messages in total. Additional network parameters can be found in Table III.

Figure 6 shows the average payload throughput for single nodes and Figure 7 the aggregated throughput of all nodes.

For 64-byte messages, the payload throughput is between 200 and 260 MB/s for all node numbers, showing a minimal decrease from 2 to 16 nodes. With 4096-byte messages the throughput improves with up to 8 nodes peaking at 1370 MB/s full-duplex bandwidth (5.5 GBit/s uni-directional). With 64 nodes the throughput is still above 5 GBit/s resulting in an aggregated throughput of 83,376 MB/s. The minor decline in both experiments can be explained by an uneven deployment of our network benchmark causing the last nodes starting and finishing a few seconds later. The end-to-end throughput between two nodes seems to be bound at around 3.2 GBit/s in the Microsoft Azure cloud as tests with iperf showed, too.

The benchmarks show that DXNet, as well as EthDXNet scale very well for asynchronous messages under high loads.

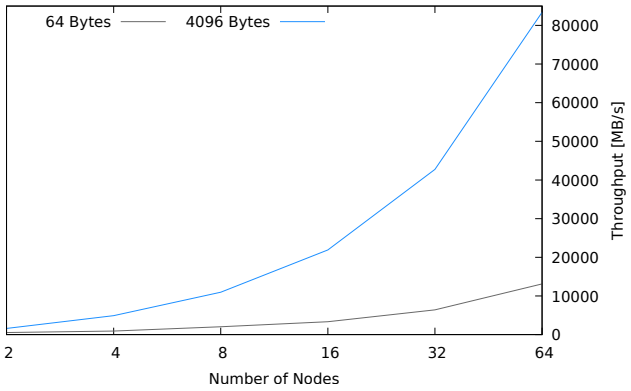


Figure 7. Aggregated Message Payload Throughput. 1 Application Thread, 2 Message Handler Threads

B. Request-Response Latency

The next benchmarks are used to evaluate request-response latency by measuring the **Round Trip Time** (RTT) which includes sending a request, receiving the request, sending the corresponding response and receiving the response. Figure 8 shows the RTTs for an all-to-all scenario with 2 to 64 nodes and 1, 16 and 100 application threads. Furthermore, all-to-all tests with `ping` are included to show network latency limitations.

The latency of the Azure Ethernet network is relatively high with a minimum of 352 µs measured with DXNet and one application thread (Figure 8). A test with up to 4032 ping processes shows that the average latency of the network is even higher (> 500 µs). In DXNet, own requests are combined with responses (and other requests if more than one application thread is used). This reduces the average latency for requests. Additionally, the ping baseline shows an increased latency for more than 32 nodes, by using one scale-set for the first 32 nodes and another one for the last 32 nodes. Different scale-sets are most likely separated by additional switches which increases the latency for communication between scale-sets.

EthDXNet is consistently under the ping baseline demonstrating the low overhead and high scalability of EthDXNet (and DXNet) when using one application thread. With 16 application threads, the latency is slightly higher and on the same level as the baseline, but the throughput is more than 10 times higher as well (in comparison to DXNet with one application thread). Furthermore, both lines have the same bend from 32 to 64 nodes as the baseline.

With 100 application threads per node (up to 6,400 in total), the latency increases noticeably, as expected, because the CPU is highly overprovisioned. In this situation the latency between writing a message into the ORB and sending it increases dramatically with more open socket channels. Furthermore, requests can be aggregated more efficiently in the ORBs with less open connections masking the overhead with a few nodes.

The latency experiments show that EthDXNet scales up to 64 nodes without impairing latency. With a very high number of application threads (relative to the available cores) the latency increases but is still good.

 ISBN: 978-1-61208-633-0

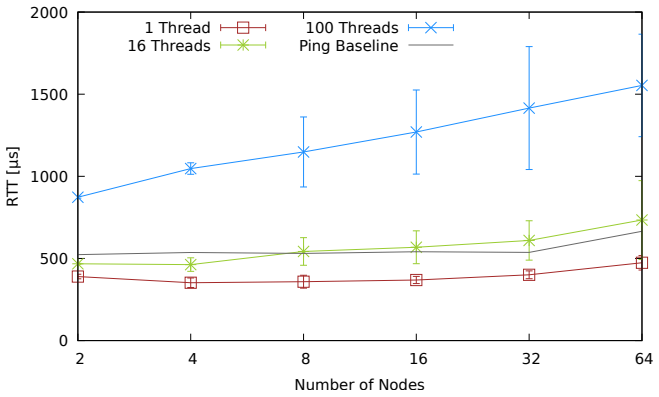


Figure 8. Average Request-Response Latency. 1 to 100 Application Threads, 2 Message Handler Threads

IX. Conclusions

Big data applications, as well as large-scale interactive applications are often implemented in Java and typically executed on many nodes in a cloud data center. Efficient network communication is very important for these application domains.

In this paper, we described our practical experiences in designing a transport implementation, EthDXNet, based on Java.nio, integrated into DXNet. EthDXNet provides a double-channel based automatic connection approach using back-channels for sending flow control data and an efficient operation interest handling which is important to achieve low-latency message handling with Java.nio's Selector.

Evaluation with micro benchmarks in the Microsoft Azure cloud shows the scalability of EthDXNet (together with DXNet) achieving an aggregated throughput of more than 83 GByte/s with 64 nodes connected with 5 GBit/s Ethernet (10 GBit/s Ethernet limited by SLAs). Request-response latency is almost constant for an increasing number of nodes as long as the CPU is not overloaded. Future work includes experiments on larger scales with application traces.

References

[1] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: Graph processing at facebook-scale," *Proc. VLDB Endow.*, vol. 8, pp. 1804–1815, Aug. 2015.

[2] S. Ekanayake, S. Kamburugamuve, and G. C. Fox, "Spidal java: High performance data analytics with java and mpi on large multicore hpc clusters," in *Proceedings of the 24th High Performance Computing Symposium*, 2016, pp. 3:1–3:8.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[4] S. Microsystems, "Java remote method invocation specification," https://docs.oracle.com/javase/7/docs/platform/rmi/spec/rmiTOC.html, accessed: 2018-03-14.

[5] Oracle, "Package java.net," https://docs.oracle.com/javase/8/docs/api/java/net/package-summary.html, accessed: 2018-03-14.

[6] S. Mintchev, "Writing programs in javampi," School of Computer Science, University of Westminster, Tech. Rep. MAN-CSPE-02, Oct. 1997.

[7] K. Beineke, S. Nothaas, and M. Schoettner, "Efficient messaging for java applications running in data centers," Feb. 2018, preprint on webpage at https://cs.hhu.de/en/research-groups/operating-systems/publications.html.

[8] S. P. Ahuja and R. Quintao, "Performance evaluation of java rmi: A distributed object architecture for internet based applications," in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '00, 2000, pp. 565–569.

[9] M. Philippsen, B. Haumacher, and C. Nester, "More efficient serialization and rmi for java," *Concurrency: Practice and Experience*, vol. 12, pp. 495–518, 2000.

[10] R. Latham, R. Ross, and R. Thakur, "Can mpi be used for persistent parallel services?" in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer Berlin Heidelberg, 2006, pp. 275–284.

[11] J. A. Zounmevo, D. Kimpe, R. Ross, and A. Afsahi, "Using mpi in high-performance computing services," in *Proceedings of the 20th European MPI Users' Group Meeting*, ser. EuroMPI '13, 2013, pp. 43–48.

[12] Oracle, "Java i/o, nio, and nio.2," https://docs.oracle.com/javase/8/docs/technotes/guides/io/index.html, accessed: 2018-03-14.

[13] W. Pugh and J. Spacco, *MPJava: High-Performance Message Passing in Java Using Java.nio*. Springer Berlin Heidelberg, 2004, vol. 16.

[14] R. Hitchens, *Java NIO*. Sebastopol, CA, USA: O'Reilly Media, 2009.

[15] G. L. Taboada, J. Touriño, and R. Doallo, "Java fast sockets: Enabling high-speed java communications on high performance clusters," *Comput. Commun.*, vol. 31, pp. 4049–4059, Nov. 2008.

[16] K. Beineke, S. Nothaas, and M. Schoettner, "Dxnet project on github," https://github.com/hhu-bsinfo/dxnet, accessed: 2018-03-14.

 ISBN: 978-1-61208-633-0

Bandwidth Scheduling for Maximizing Resource Utilization in Software-Defined Networks

Poonam Dharam
Department of Computer Science
Saginaw Valley State University
Saginaw, MI, USA
Email: pdharam@svsu.edu

***Abstract*—Software-Defined Networks (SDNs) enable the network control plane to be decoupled from the data plane and assign the control to a programmable software unit, i.e., controller. With such a logically centralized control, SDNs provide the capability of bandwidth reservations that meets user requirements. In SDNs with multiple logically centralized controllers, it is challenging to maintain accurate link-state information at every controller in a consistent manner. Bandwidth scheduling in presence of inaccuracy may lead to rejection of reservation requests in turn affecting the Quality of Service (QoS) provided by the Network Service provider. In order to minimize the service disruption caused by lack of uniform global network view (GNV) at controllers, we propose a routing scheme based on multiple weights' distribution such that the number of requests dropped or rejected during bandwidth reservation setup phase is minimized. The performance superiority of the proposed bandwidth reservation solutions is illustrated by extensive simulations in comparison with existing methods.**

Keywords*: *Software-Defined Networks; bandwidth reservation; bandwidth scheduling.

I. INTRODUCTION

Next-generation e-science applications in various domains generate colossal amounts of simulation, experimental, or observational data, on the order of terabyte at present and petabyte or exabyte down the road, now frequently termed as "Big Data", which must be transferred to remote sites for collaborative processing, analysis, and storage. The data transfer at such scales necessitates the development of high-performance networks that are capable of provisioning dedicated channels with reserved network resources through either circuit/lambda-switching or Multi-Protocol Label Switching (MPLS) tunneling techniques.

The decoupling of control and data planes in SDNs gives network designers freedom to re-factor the network control plane, allowing the network control logic to be designed and operated as though it were a centralized application, rather than a distributed system. SDNs with logically centralized control and distributed flow tables enable various networking services, such as bandwidth reservations to support Big Data transfer. Such reservations not only ensure the confirmed availability of bandwidth during the entire course of data transfer, but also yield a higher level of resource utilization in the network. In addition to Big Data transfer, some real-time applications, such as Video-on-Demand (VoD) or virtual collaboration might require an immediate allocation of bandwidth for an indefinite duration [1].

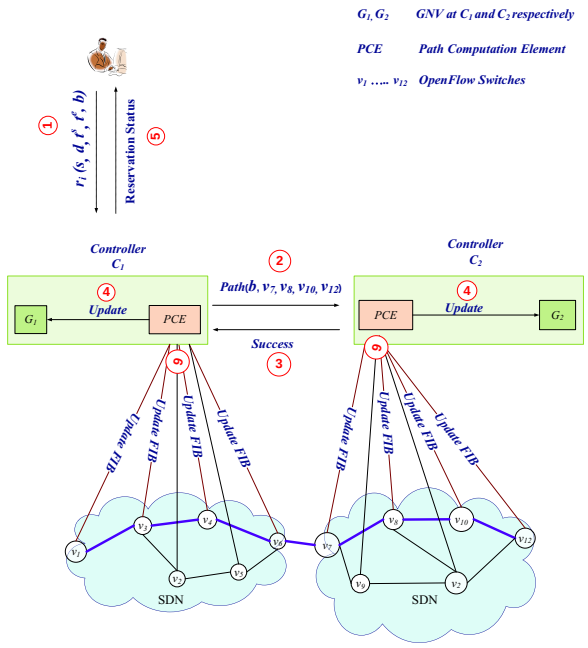


Fig. 1. Bandwidth scheduling in Software-Defined Networks.

We now describe the process of scheduling bandwidth reservation requests in SDNs using a simple example shown in Figure 1. The bottom layer consists of OpenFlow switches, v_1, v_2, ..., v_{12}, grouped into one or more separate domains, responsible for forwarding incoming packets based on the information stored in their Forward Information Base (FIB) consisting of flow entries and next-hop information. C_1 and C_2 are controllers consisting of Global Network Views (GNVs) G_1 and G_2, respectively, consisting of upto-date link-state information of the underlying network. The Path Computation Element (PCE) is responsible for processing the incoming requests for path setup and updating the flow entries in FIB. The controllers exchange their link-state information among themselves to maintain a consistent GNV across every controller.

When a bandwidth reservation request $r_i(s, d, t^s, t^e, b)$ arrives at controller C_1, the controller follows a two-phase process. In Phase 1, the controller checks its GNV to verify if b units of bandwidth are available. If the network has sufficient

 ISBN: 978-1-61208-633-0

bandwidth, controller C_1 computes a path $p(r_i)$ that connects source s and destination d and meets the bandwidth requirement of b units for the time duration [t^s, t^e], according to the advertised link-state information. From the perspective of the controller, if there is sufficient network resource (bandwidth) to accommodate the request, it launches a signalling/setup process to ensure that b units of bandwidth are indeed available on every link $l \in p(r_i)$ during [t^s, t^e]. As the signalling message traverses all the component links along the selected path $p(r_i)$, each responsible controller performs an admission test to check if a component link in its domain can actually support the request. If the link has sufficient resource, the controller reserves bandwidth on behalf of the new connection before forwarding the setup message to the controller servicing the next component link along the path; otherwise, it rejects the connection request immediately. Upon the confirmation of the user request, the controllers employ the state distribution method to distribute their state information to other controllers in an attempt to achieve a consistent GNV across the network.

When the link-state information in the GNV is not up-to-date, i.e., it does not represent the current picture of the network state, the routing process might select a path that is unable to support the call requirements. Consequently, a reservation request that is initially scheduled on a particular path would be rejected in the path setup process, which is termed as a false positive. Out-of-date information may also cause controllers to make a wrong decision on path selection; some paths may be overused (since the GNV is updated periodically, the routing scheme tend to select the same path for an extended period of time [2]) and hence cannot meet the required QoS while there exist other underutilized paths. This is widely recognized as a routing inaccuracy problem. The false positive resulted from inaccuracy jeopardizes users' satisfaction [3]. This situation deteriorates as the level of inaccuracy/inconsistency increases.

The setup failure in a blocked or rejected request incurs extra overhead to reserve resources along the path and may delay the establishment of other connections. In addition, a failed connection temporarily holds resources on its upstream links, which may block other connections in the interim [4] [5]. Also, the performance of a QoS routing algorithm can be significantly undermined by inaccurate link-state information. Thus, it is critical to minimize the setup failures caused by inaccurate state information for efficient network utilization and improved QoS. A good routing algorithm must incorporate mechanisms to account for the inaccuracy in the GNV and make effective routing decisions in the presence of such inaccuracy [6].

We propose a weight-based routing scheme, called MinBlock-Routing, to schedule incoming bandwidth reservation requests such that the total number of reservation requests blocked due to inaccurate GNV is minimized. When multiple requests with similar requirements (source, destination, bandwidth, start-time, and end-time) arrive at controllers simultaneously, our routing scheme assigns multiple weights to links connecting the same pair of OpenFlow switches, but belonging to different controllers, thus choosing different paths for similar requests.

The rest of the paper is organized as follows. Section II conducts a brief survey of related work. Section III constructs the cost models and formulates the bandwidth reservation problem under study. Section IV details the scheduling algorithm design. Section V evaluates the scheduling performance in simulated networks.

II. Related Work

Most of the work in this field has been focused on QoS routing whose goal is to find a path that satisfies multiple QoS constraints while maximizing network utilization. The existing work on this subject can be broadly classified into single-path routing and multi-path routing.

In single-path routing, only a single path is considered between a source-destination pair. Some work in this category only considers bandwidth in path computation. In [7], Guerin *et al.* proposed source routing algorithms with inaccurate link-state information by exploring the impact of information inaccuracy in the context of QoS routing. In [2], Apostolopoulos *et al.* proposed a new routing mechanism, Safety Based Routing (SBR), to address the routing inaccuracy issues when computing explicit paths with bandwidth constraints. SBR incorporates a new link attribute, safety (S), which represents the effect of the routing inaccuracy in the link-state reliability, in the path selection process. In [4] [8] Bruin *et al.* proposed a QoS routing mechanism, BYPASS Based Routing (BBR), which bypasses those links along the selected path that potentially cannot satisfy the traffic requirements.

There exists some other work that considers both bandwidth and delay in path computation. In [9], Korkmaz *et al.* addressed the path computation problem under bandwidth and delay constraints with inaccurate link-state information. They adopted a probabilistic approach where the state parameters are characterized by random variables to find the most-probable bandwidth-delay-constrained path (MP-BDCP). In [10], Zhang *et al.* studied QoS routing in the presence of inaccurate state information, formulated as the Most-Probable Two Additive-Constrained Path (MP-TACP) problem. They proposed a routing algorithm that uses pre- and on-demand computation along with both linear and nonlinear search techniques that take inaccuracies into consideration. In most existing bandwidth scheduling algorithms, reservation requests with similar requirements (the same source/destination and overlapped start/end time) are typically routed along the same path, hence resulting in unbalanced traffic. This situation becomes worse in the presence of inconsistent GNVs in SDNs.

Multi-path routing, on the other hand, probes multiple feasible paths simultaneously, and if more than one path can satisfy the QoS requirement of a request, the shortest one is typically selected. In [11], Jia *et al.* proposed a routing strategy, in which connection requests with specific bandwidth demands can be assigned to one of several alternative paths. They introduced a collection of k-shortest path routing schemes and investigated

 ISBN: 978-1-61208-633-0

the performance under a variety of traffic conditions and network configurations. The idea of randomized routing [2], is to compute a set of feasible paths and then randomly select one for a connection request. Every time when the link-state is updated, k paths with the largest bandwidths are selected as the candidates to be used until the next link-state update. When a request arrives, it is randomly routed among the k paths. Since the routing process does not always select the best path, the requests can be distributed across multiple available paths for better load balance. In [12], Dharam *et al.* proposed a randomization-based scheme for path computation by assigning random weights to links during path computation, in turn distributing the incoming requests across multiple path Existing schemes probe multiple paths either concurrently, which introduces much higher overhead than their single-path counterparts, or sequentially, which incurs a longer path setup time. Similar to multi-path routing, randomized path selection also requires computing a set of candidate paths to the destination, which incurs overhead for maintaining the information of all the computed paths.

Our proposed routing scheme distributes incoming reservation requests with similar requirements, in terms of source and destination, across multiple paths such that the time taken to update the GNVs does not affect the ongoing routing decisions.

III. Cost Models and Problem Formulation

In this section, we discuss the cost models by using a graph to represent the SDN and formulate our problem based on the assumptions made.

A. Cost Models

We use a network graph $G_j[V, E, B]$ to represent the latest GNV at controller C_j, where V is a set of OpenFlow switches, E is a set of network links, and B is a set of advertised bandwidths $b_l^{ad}(t)$ for each link l in the network at a future time point t.

For a bandwidth reservation request $r_i(s_i, d_i, t_i^s, t_i^e, b_i)$ arriving at a controller C_j, the controller needs to compute a path $p(r_i)$ that connects source s_i and destination d_i and meets the requirement of bandwidth b_i for a time duration $[t_i^s, t_i^e]$, according to the advertised link-state information. From the perspective of the controller, if there is sufficient network resource (bandwidth) to accommodate the request, it launches a signalling/setup process such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) to ensure that b_i units of bandwidth are indeed available on every link $l \in p(r_i)$ during $[t_i^s, t_i^e]$. As the signalling message traverses all the component links along the selected path $p(r_i)$, each responsible controller performs an admission test to check if a component link in its domain can actually support the request. If the link has sufficient resource, the controller reserves bandwidth on behalf of the new connection before forwarding the setup message to the controller servicing the next component link along the path; otherwise, it rejects the connection request immediately. Upon the confirmation of the user request, the controllers employ the state distribution method to distribute their state information to other controllers in an attempt to achieve a consistent GNV across the network.

A link without sufficient available bandwidth along the path computed based on the advertised bandwidth would lead to a setup failure, and the inaccuracy in GNV is one major cause for such setup failures.

B. Problem Formulation

We formulate a Bandwidth Reservation (MinBlock-Routing) problem as follows:

Definition 1: **MinBlock-Routing** Given an SDN with multiple controllers, a graph $G_j[V, E, B]$, which represents the latest GNV at controller C_j and a set $R_j = \{r_1, ..., r_i, r_{i+1}, ..., r_n\}$ of n bandwidth reservation requests at C_j, where each request $r_i \in R_j$ specifies source s_i, destination d_i, start time t_i^s, end time t_i^e, and required bandwidth b_i, we propose a bandwidth scheduling scheme with the goal of minimizing the total number of requests blocked at the time of resource reservation, due to inconsistent GNVs.

IV. Bandwidth Scheduling Algorithm

We first generate a new network topology G' from G_j by pruning the links from G_j without sufficient bandwidth during the time interval $[t_i^s, t_i^e]$ to accommodate the incoming request r_i due to the existing bandwidth reservations within that interval. Multiple weights are assigned to links that connect OpenFlow switches belonging to same pair of controllers. For example, let link l_1 connect two OpenFlow switches sw_{11} and sw_{21} such that sw_{11} and sw_{21} belong to controllers C_1 and C_2 respectively. Also, let link l_2 connect two OpenFlow switches sw_{12} and sw_{22} such that sw_{12} and sw_{22} belong to C_1 and C_2 respectively. Thus, there are two possible ways of connecting C_1 and C_2. By forcing C_1 to use l_1 and C_2 to use l_2 to move traffic between the two SDN networks, the GNV inconsistency due to inaccurate state information can be minimized. Both C_1 and C_2 agree upon randomly generated weights for l_1 and l_2 as follows: C_1 assigns a lower weight to l_1 and higher weight to l_2 where as C_2 does the reverse i.e., assigns a higher weight to l_1 and lower to C_2. For a path from C_1 to C_2, l_1 is chosen, whereas for a path from C_2 to C_1, l_2 is chosen. Thus, by assigning multiple weights, controllers tend to choose multiple paths between the same set of networks. This in turn gives time for the controllers using same path to update their GNVs, thus reducing the inconsistent view. If a valid path exists for a reservation request, then bandwidth is reserved along the path and the GNV is updated with the new bandwidth values; otherwise, if no valid path is found for a reservation request, then the reservation request is rejected.

This algorithm is executed by a top-level controller with the network and request information in each network domain. If such a global controller does not exist, the algorithm could be executed in a distributed manner on each controller with its local information and the remote information sent by its peers.

V. Performance Evaluation

The simulations are executed in a set of randomly generated networks comprised of 10 Gbps links with varying sizes from small to large scales and a set of hundreds of reservation

 ISBN: 978-1-61208-633-0

TABLE I
THE SIZES OF NETWORKS USED IN THE SIMULATIONS.

Index of problem sizes	# of nodes	# of links
1	5	12
2	10	20
3	15	34
4	20	42
5	25	51
6	30	65
7	35	74
8	40	88
9	45	91
10	50	102
11	55	120
12	60	134

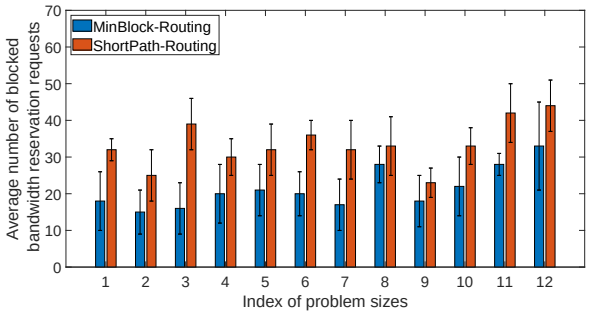


Fig. 2. The average number of blocked reservation requests with the standard deviations resulted from ShortPath-Routing and MinBlock-Routing.

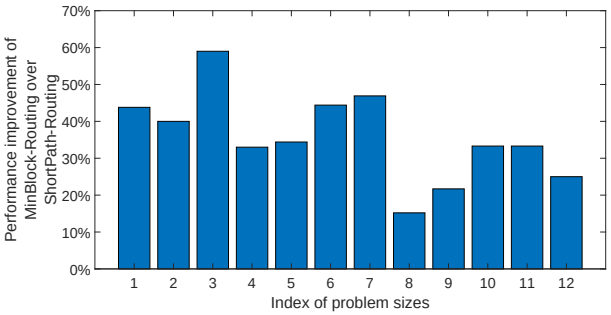


Fig. 3. The performance improvement of MinBlock-Routing over ShortPath-Routing in terms of percentage reduction in blocked bandwidth reservation requests.

requests under Poisson distribution with random input parameters chosen within appropriate ranges.

We consider 12 network sizes, indexed from 1 to 12, each with a different number of nodes and links as tabulated in Table I. For each network size, we generate 10 random problem instances of different network topologies. We run the proposed MinBlock-Routing algorithm and also a traditional scheduling shortest path routing algorithm (ShortPath-Routing), for comparison as it has been widely used for bandwidth reservation in real-life high-performance networks.

We evaluate the global routing performance in large-scale multi-domain networks with multiple controllers, each of which processes around 100 bandwidth reservation requests. The proposed MinBlock-Routing algorithm is able to balance the loads of incoming requests by distributing them across various alternative paths, and hence significantly reduce the number of blocked requests. Figure 2 shows the average number of blocked reservation requests with the standard deviations resulted from ShortPath-Routing and MinBlock-Routing. For a visual comparison, we further plot in Figure 3 the performance improvement of MinBlock-Routing over ShortPath-Routing in terms of the percentage decrease in blocking of incoming reservation requests, calculated as: $\frac{B(ShortPath)-B(MinBlock)}{B(ShortPath)} \cdot 100\%$. We observe that the proposed MinBlock-Routing algorithm significantly outperforms ShortPath-Routing. Our proposed Routing algorithm is able to route the incoming reservation requests with similar requirements across multiple paths in presence of inaccuracies in GNVs.

VI. CONCLUSION

We formulated and solved a bandwidth scheduling problem for minimizing the number of reservation requests blocked in software-defined networks with multiple controllers and inconsistent GNVs. In the current work, we consider scheduling incoming reservation requests with similar requirements across multiple paths thus minimizing the blocking ratio of user requests for bandwidth reservation.

REFERENCES

[1] I. Ahmad, J. Kamruzzaman, and S. Aswathanarayaniah, “A book-ahead routing scheme to reduce instantaneous request call blocking and preemption rate,” in Proc. of IEEE ICON, vol. 1, Nov. 2005, p. 6.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, “Improving qos routing performance under inaccurate link state information,” in Proceedings of the 16th International Teletraffic Congress, 1999.

[3] N. Ansari, G. Cheng, and N. Wang, “Routing-oriented update scheme (rose) for link state updating,” in IEEE Transactions on Communication, 2008.

[4] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, and J. Domingo-Pascual, “Optimizing routing decisions under inaccurate network state information,” in Lecture Notes in Computer Science No.3375: Quality of Service in Multiservice IP Networks, 2005, pp. 445–455.

[5] A. Shaikh, J. Rexford, and K. G. Shin, “Evaluating the impact of stale link state on quality-of-service routing,” IEEE/ACM Transactions On Networking, vol. 9, 2001.

[6] X. Yuan, W. Zheng, and S. Ding, “A comapritive study of qos routing schemes that tolerate imprecise state information,” in Intl. Conf. on Computer Comm. and Netw., Oct. 2002, pp. 230 – 235.

[7] R. Guerin and A. Orda, “Qos routing in networks with inaccurate information: Theory and algorithms,” IEEE/ACM Trans. on Networking, vol. 7, no. 3, pp. 350–364, 1999.

[8] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, and J. Domingo-Pascual, “Qos routing algorithms under inaccurate routing information for bandwidth constrained applications,” in Proc. of Int. Conf. on Communications, vol. 3, 2003, pp. 1743 – 1748.

[9] T. Korkmaz and M. Krunz, “Bandwidth-delay constrained path selection under inaccurate state information,” IEEE/ACM Transactions On Networking, vol. 11, no. 3, 2003.

[10] Y. Zheng and T. Korkmaz, “Two additive-constrained path selection in the presence of inaccurate state information,”J. Comp. Comm., vol. 30, pp. 2096 – 2112, Jun. 2007.

[11] Y. Jia, I. Nikolaidis, and P. Gburzynski, “On the effectiveness of alternative paths in qos routing: Research articlesemphIntl. Journal of Communication Systems, vol. 17, 2004.

[12] P. Dharam, C. Q. Wu, and N. S. V. Rao, “Advance bandwidth scheduling in software-defined networks,” in Globecom, 2015.

 ISBN: 978-1-61208-633-0