Computers and Electrical Engineering 000 (2016) 1–14

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

An evolvable, scalable, and resilient control channel for software defined wireless access networks☆

Kien Nguyen*, Kentaro Ishizu, Fumihide Kojima

National Institute of Information and Communications Technology 3–4 Hikarinooka, Yokosuka, Kanagawa, Japan 239–0847, Japan

ARTICLE INFO

Article history:
Received 25 February 2016
Revised 13 September 2016
Accepted 13 September 2016
Available online xxx

Keywords:
Openflow
SDWAN
Control channel
Scalability
Resilience
Evolvability

ABSTRACT

This paper presents a novel multipath communication-based OpenFlow channel for Software Defined Wireless Access Networks (SDWANs), namely mOpenFlow. The advantageous features of mOpenFlow include the following: (i) resilience and scalability in wireless environments, (ii) evolvability of the existing access networks and the OpenFlow standard, (iii) a novel network calculus-based model for performance analysis of mOpenFlow. By leveraging the multipath communication for conveying OpenFlow traffic, mOpenFlow enhances both robustness (i.e., resilience) and throughput (i.e., scalability) of the control channel. To achieve the evolvability, mOpenFlow adopts the multipath transport control protocol, which conforms to SDWANs and the OpenFlow standard. We evaluate mOpenFlow in an emulated SDWAN in relation to the standard channel. The results show that mOpenFlow outperforms the standard channel, both in terms of robustness and scalability. Additionally, the numerical results indicate that the model provides a fast and reliable way for analyzing the end-to-end delay on mOpenFlow.

1. Introduction

Software Defined Networking (SDN) is recognized as one of the key technologies contributing to the proliferation of mobile device adoption, which requires significant improvements in network performance and network control [1]. Accordingly, many improvements are proposed for Software Defined Wireless Access Networks (SDWANs) [2–4]. In an SDWAN, an SDN controller logically centralizes control functions and manages distributed networking devices (e.g., middleboxes, packet gateways, mobile devices, etc.). The devices and the controller exchange the control and update information on the control channel (i.e., OpenFlow channel), which is the most important component of the SDWAN.

As the control channel has to deal with novel requirements, it must be robust and scalable. However, the standard OpenFlow, which was originally designed for wired networks, proves inefficient when applied directly in SDWANs. Specifically, the SDWAN's control traffic is indeed conveyed on both wired and wireless infrastructures, which may contain a mix of inband and out-of-band deployments. Those specific deployments are not fully specified in the OpenFlow standard [5]. On the other hand, when the standard channel experiences failure, the SDN devices could not work properly (i.e., fall into fail modes). Consequently, severe damages or huge data loss may occur in the SDWANs. Therefore, it is of great importance to design a new control channel that conforms to the requirements of the SDWAN and the standard.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. S. Smys.

* Corresponding author.
E-mail addresses: kienng@nict.go.jp (K. Nguyen), ishidu@nict.go.jp (K. Ishizu), f-kojima@nict.go.jp (F. Kojima).

http://dx.doi.org/10.1016/j.compeleceng.2016.09.014
0045-7906/

This work first analyzes the disadvantages of standard OpenFlow, and then addresses the requirements necessary for the control channel in SDWANs (i.e., robustness, resilience, and evolvability). We then propose the novel OpenFlow channel, namely mOpenFlow[1], which appropriately satisfies the requirements. The following are the advanced features of mOpenFlow:

- mOpenFlow achieves robustness and scalability by leveraging its advantage of multipath communications.
- mOpenFlow evolves seamlessly on the mix deployment and the OpenFlow standard by using the transport layer multipath's solution (i.e., the standard multipath TCP (MPTCP)).
- mOpenFlow is modeled by a newly proposed framework, which is based on the theory of network calculus.

To validate the effectiveness of mOpenFlow (mOF), we evaluate its performance in an emulated SDWAN, in relation to that of the standard OpenFlow channel (sOF). The results show that mOF outperforms sOF in terms of both robustness and scalability. On the other hand, the theoretical model and analytical results show that the proposed framework is capable of modeling mOpenFlow and revealing its performance bounds.

For the scalability and robustness of OpenFlow channel, the focus of previous works has been mostly on designing distributed SDN controllers [7]. In contrast to this, our design targets the case of a single controller. However, our design can be integrated with the existing ones to further improve the channel performance. Other works that focus on improving the scalability depend on either devolving control functions to SDN devices or using multi-thread controllers [8]. For robustness improvement also some contributions are available, but they are for the OpenFlow wide area network [9] or require the changes of OpenFlow standard [10]. Those approaches can not bypass the requirement of evolvability. We reckon our work is unique in that it considers the scalability, robustness, and evolvability of OpenFlow channel of SDWAN.

The rest of paper is organized as follows. Section 2 briefly reviews related works. Section 3 presents the requirements of control channel in SDWANs and the proposed mOpenFlow channel. Section 4 includes the approach of modeling mOpenFlow. Section 5 deals with the evaluation method and results. Finally, Section 6 sums up the conclusions drawn from this study.

2. Related works

The explosive increase in mobile devices will result in a huge demand for usage data, various levels of QoS and mobility. Hence, the existing wireless access networks should be upgraded to cope with the novel levels of dynamic and scalability. The fast evolving SDN technology shows great potential for improvement of the access networks. SDN has been proven to be efficient in the wired network for scalability and resilience [11,12]. In the wireless domain, there have been increasing SDWAN proposals, which aim to solve new challenges in future access networks.

In [2], SDN shows its efficiency in handling mobility in an SDWAN including Wi-Fi and WiMax. In [3], the authors show that SDN is the key technology in bypassing the current limitations in large scale cellulars (i.e., path inflation, lack of routing function, lack of scalability, vendor-dependency). In [4], a novel SDN-based architecture has been proposed to provide an open platform for making innovations on wireless access networks. These works use the standard OpenFlow as a southbound interface in SDWANs. However, its robustness and scalability have been seldom discussed.

The focus of previous works on the robustness and scalability of the standard channel has been mainly on wired networks. The common solutions include introducing multiple controllers and designing a scalable controller [7,9]. The OpenFlow standard also specifies a resilience method for the channel by using a secondary controller. In [10], to effectively deal with the failure of OpenFlow channel, the authors propose a new type of message and a failure handling method, which requires adding more intelligence to SDN devices. The method can be used in SDWAN but requiring modification of OpenFlow standard.

Our approach is unique in that it aims at designing a novel channel, which is scalable, robust, and evolvable for SDWANs. The mOpenFlow channel leverages multipath communications for scalability and robustness. mOpenFlow selects the standard multipath transport layer (i.e., MPTCP) for the sake of compatibility with both the OpenFlow standard and the physical deployment of SDWANs. Along with increasing popularity of SDN, efforts for improving its theoretical performance analysis have also been increasing concurrently. Regarding the OpenFlow channel's analysis, in [13] the authors use queuing theory to investigate the performance of OpenFlow architecture. Although the work inspires others to explore the domain of performance analysis in SDN, it still shows some limitations. The Markovian servers are assumed for both SDN controllers and SDN devices. Moreover, it considers only the standard OpenFlow channel. The authors of [14] initially propose to use the network calculus theory for modeling and analyzing SDN components. They show that the basic models of network calculus provide a convenient way to find the bound performance values in SDN. Compared to the previous work, our analysis is for a newly proposed multipath OpenFlow channel. We extend the basic model to capture the state of the new channel.

[1] The first version of mOpenFlow has been published in [6]. In this version, the modeling and theoretical analysis, as well as, numerical results parts are added.

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

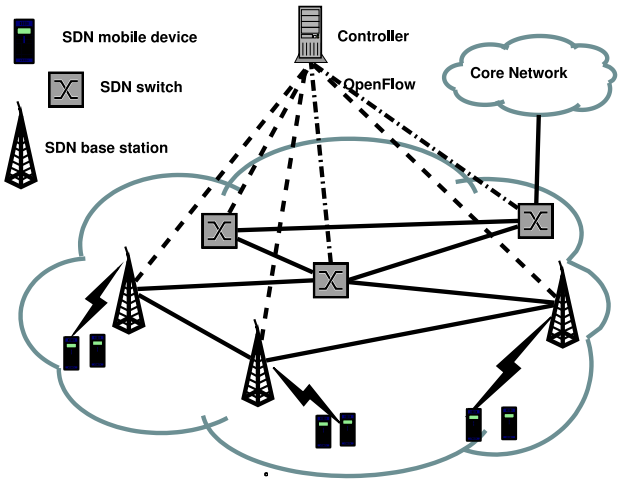

Fig. 1. Example of SDWAN.

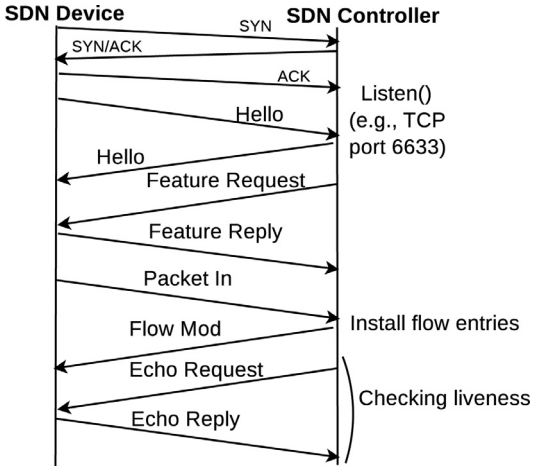

Fig. 2. Communication on the standard channel.

3. Designing a novel control channel for SDWAN

3.1. Overview

3.1.1. SDWAN

The wireless access network represents a group of networking devices, which connect to the Internet core and provide access to wireless devices. A typical access network can be seen at a Long Term Evolution (LTE)'s eNodeB, a WiMax base station, etc. In this work, SDWAN is an enhanced wireless access network to which decoupling architecture of SDN is applied. The typical SDWAN comprises an SDN controller, which manages distributed SDN (i.e., OpenFlow capable) devices. The devices could be smart phones, middleboxes, S-Gateway, P-Gateway, or Layer2/Layer3 switches at the base stations. Fig. 1 is an illustration of SDWAN's architecture, in which an SDN controller manages the devices via logically centralized manner. The controller has a consistent view of the network and communicates with the devices via the OpenFlow protocol. The forwarding policies on devices could be dynamically changed, updated, or removed only by modifying the controller's software. SDWAN is increasingly popular because SDN significantly reduces capital and operational expenditure. Moreover, SDN evolves not only on the existing access networks, but also holds promise to support many new features. For example, challenging problems such as slicing resources or sharing infrastructure between operators will become practical in SDWANs.

3.1.2. The standard OpenFlow channel

A typical communication on the standard OpenFlow is illustrated in Fig. 2. The SDN controller always listens to incoming connections (e.g., via TCP port 6633) from SDN devices. The communication is initiated by the TCP handshake (exchanging

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

SYN/(SYN/ACK)/ACK), followed by the OpenFlow handshake (i.e., exchanging a pair of *Hello* messages). After the initiation, the controller sends the *Feature Request* message and receives the *Feature Reply* message to get the device's information. The device maintains flow tables, each of which contains forwarding rules for arriving matched packets (i.e., flow entries). A matched packet will be processed by an action (i.e., drop, modify header, etc.) or a chain of actions. When there is no matching rule for an arriving packet, the packet is encapsulated in a *Packet In* message and sent to the controller. The controller creates an appropriate rule and populates the device with a *Flow Mod* message. The message also indicates as to how long the device needs to store the matched entry. The storing period could be indefinite or fixed. Besides that, there is a liveness checking mechanism for the channel status, which represents a periodical exchange of the *Echo Request*/*Echo Reply* messages.

The standard OpenFlow defines two types of physical deployment for the OpenFlow messages: inband network and out-of-band network. In the former, the OpenFlow traffic and data traffic share the same infrastructure, whereas in the latter, the OpenFlow traffic needs an extra network for transferring. The standard also requires TCP/IP for all data transfer on the OpenFlow channel.

3.2. Requirements of control channel in SDWAN

The standard OpenFlow, though provides a rich set of new features, is not well suited to the physical deployment of SDWAN. This is because SDWAN includes a wide range of SDN devices (e.g., fixed and mobile devices, etc.). In a SDWAN with the standard channel, the OpenFlow traffic is on the mix network that includes both out-of-band (e.g., for SDN switches) and inband connections (e.g., for mobile devices). Unfortunately, the standard is yet to well define the mentioned case. Therefore, a novel channel is expected to be suitable to the new physical deployment (i.e., mix), besides being backward compatible with the OpenFlow standard.

Another important issue to consider is the channel's robustness. Because of the requirement of TCP/IP, the OpenFlow traffic is actually exchanged on an IP-based path (i.e., route) between the device and the controller. In SDWANs, the IP-based path may incur failure due to many reasons, such as failed physical devices, mobility, etc. In the such cases, the network has to wait until reaching a stable routing state before continuing to transmit data. In a complex network, the long convergence of IP routing protocols (e.g., OSPF) may drive the network into unexpected states. Moreover, if the SDN device is not appropriately controlled it immediately falls into the fail standalone or fail secure modes. In the latter mode, the device keeps the running configuration, but discards all the OpenFlow messages. Meanwhile, in the former one, the device behaves only as an Ethernet switch. Additionally, the SDN device may remove flow entries after predetermined timeout values.

On the other hand, SDWAN is expected to handle future scenarios with a large number of devices and quality of service demands. Undoubtedly, the number of requests generated from devices will sharply increase and accordingly, the responses generated from controller increase appropriately. Hence, the throughput of control channel is expectedly scalable. Besides that, the IP-based path caps the throughput of channel (i.e., the path capability) also. A new control channel expectedly overcomes these disadvantages and achieves scalability in an effective manner.

3.3. mOpenFlow: a multipath OpenFlow channel for SDWANs

3.3.1. Enhancing robustness

We propose to use multipath communications for enhancing robustness, the most important requirement of OpenFlow channel. Based on the work in [15], we can theoretically represent the robustness of OpenFlow channel via the availability of the paths between a controller and an SDN device. Obviously, the larger the value of availability the higher is the level of robustness. Assuming that n paths are available for the OpenFlow channel, we denote the average availability of the path i as A_i. A_i can be calculated using the mean of uptime (m_i^{up}) and the mean of downtime (m_i^{down}) as in (1).

$$A_i = \frac{m_i^{up}}{m_i^{up} + m_i^{down}} \tag{1}$$

If the OpenFlow traffic can be concurrently transmitted over multiple paths, the OpenFlow channel fails only if all the paths fail. The statement is represented by (2).

$$A_{OF} = 1 - \prod_{i=1}^{n}(1 - A_i) \tag{2}$$

To compare the robustness of the standard channel (i.e., sOF) with that of the multipath one (i.e., mOF), we plot the correlation between the number of paths and the path availability (see Fig. 3). In the figure, sOF ($n = 1$) represents the standard OpenFlow channel, and mOF ($n = 2, 3$) the two and three paths for the OpenFlow channel, respectively. One can easily notice the robustness advantage of the mOF over sOF.

Using multipath communications, the OpenFlow traffic can be exchanged in two ways. The first one is by concurrently striping the traffic over the multiple available paths, the second one is by duplicating OpenFlow messages and sending the same messages over several paths. When the controller and the device receive several similar messages, they use the earliest arriving one and drop the remaining ones. The second method could reduce the delivery latency as proved in [16], but it is not scalable. This work focuses on the first method with evolvable technologies.

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

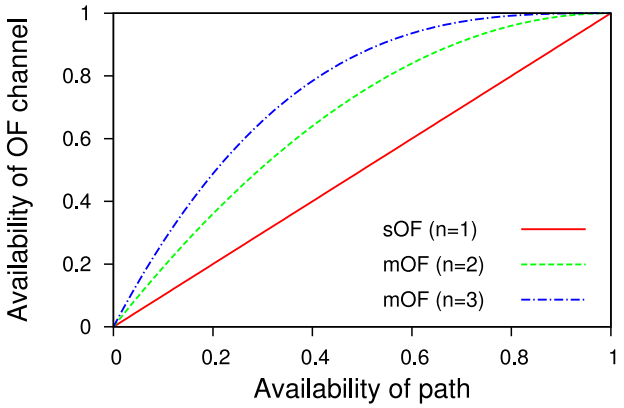

Fig. 3. Comparing the standard OF with the mOF channel.

3.3.2. Evolvability and scalability

Generally, multipath technology can achieve scalability by link/path aggregations. However, owing to the requirement of evolvability, it is necessary to investigate the available standard multipath technologies for the channel. There have been various layer 2 and layer 3 technologies such as the link aggregation with link aggregation control protocol (LACP) [17], TRILL [18], Shortest Path Bridging (SPB) [19], or Equal Cost Multipath Protocol (ECMP) [20]. However, those technologies are suitable only with a part of SDWAN, or some type of physical deployment for the OpenFlow traffic. Specifically, LACP is for Layer2 switch-to-switch's communication. LACP requires a shared IP address in one point of all the aggregation paths. SPB, ECMP and TRILL are mainly for multipath routing in data center. While TRILL requires a knowledge of entire network, SPB and ECMP relies on routing information to determine multiple best "next hops". All of the technologies have been designed for the network with Ethernet in layer 2. Besides, they requires all the related neighboring devices are equipped same technologies. Since the SDWAN may contain various types of devices, on which the technologies either impossibly works or significantly needs efforts to evolve. In our design, we follow the approach at higher layer for the mOpenFlow, which achieves multipath communication in the transport layer.

Among the multipath transport technologies, the multipath TCP is the most suitable one [21]. The technology can be evolved to be friendly with the OpenFlow standard without any modification (i.e., satisfying the TCP/IP requirement). That is because MPTCP is designed to work wherever TCP can function with only one requirement, namely the existence of MPTCP kernels at the two ends of a TCP connection. Fortunately, the MPTCP has been standardized by Internet Engineering Task Force (IETF) [22]. Moreover, the MPTCP kernel has been available on almost popular platforms in SDWANs such as Linux kernels, Android, and Apple iOS. Therefore, the requirement of using mOpenFlow can be easily fulfilled. To use mOpenFlow, the SDN devices and the controller have to just enable MPTCP.

3.3.3. mOpenFlow's operation

The operation of mOpenFlow is as follows. After the TCP handshake, as in the standard, the controller and the SDN device check for the MPTCP capable options before doing the OpenFlow handshake. Therefore, mOpenFlow needs additional delay (i.e., the average round trip time between the two components) to initiate the communication on the channel. If the MPTCP option proves incapable (e.g., on SDN devices), then mOpenFlow operates just as the standard channel does. Hence, mOpenFlow has no impact on existing MPTCP-incapable devices. If the MPTCP options are capable, mOpenFlow discovers the number of paths available between the two components. Depending on the expected configurations in the SDWAN, all the paths or one of their subsets is used for mOpenFlow. In mOpenFlow, the standard socket API (i.e., available in MPTCP) is used to interface with the OpenFlow messages. When the mOpenFlow channel uses n ($n > 1$) paths, the OpenFlow traffic is divided into n TCP flows, each of which goes on one IP-based path. All those TCP flows are finally integrated at the SDN device in the case of control messages or at the controller in the case of information messages. By using multiple paths, the achievable throughput of the channel increases, thus satisfying the scalability requirement. On the other hand, if a path experiences failures, mOpenFlow switches the traffic on the failed path to the remaining live paths. The mOpenFlow channel, therefore, is more robust than the single path channel.

4. Modeling mOpenFlow and analysis

4.1. Overview of network calculus

Network calculus is a theory based on min-plus algebra to quantitatively or qualitatively analyze the network flow's performance. The theory has been proven to be useful for analysis of performance bounds, such as delay, effective bandwidth, etc. In the general network calculus solution, arrival curves and service curves represent the input and output of

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

Table 1
NOTATION.

Symbol		Definition
$R(t)$	:	Accumulated traffic of mOpenFlow
σ	:	Burst factor of regulated mOpenFlow's traffic
ρ	:	Rate factor of regulated mOpenFlow's traffic
σ_i	:	Burst factor of path i
ρ_i	:	Rate factor of path i
d_i	:	Queuing delay on path i
f_i	:	Fixed delay on path i
D_i	:	Total delay on path i
D_{e2e}	:	End-to-end delay
i	:	Index value for path
m	:	Total number of path

investigated systems. Relying on the constraints of input and output, the performance bounds will be deduced. The network calculus-applicable domain has been expanding, ranging from the tradition network to the emerging one such as the SDN. This section introduces the basic concepts of network calculus, which are used in modeling mOpenFlow. A more detail introduction could be found in [23].

Definition 1. Cumulative function
The cumulative function $R(t)$ is defined as the number of bits on the flow in the time interval $[0, t]$, $R(0) = 0$. R being a wide-sense function.

Definition 2. Arrival curve
Given the wide-sense function α for $t \geq 0$, if the cumulative of flow $R(t)$ satisfies $R(t) - R(s) \leq \alpha(t - s)$ (i.e., $R \leq R \otimes \alpha$, $\otimes$ represents the operation of min-plus convolution), then α is said to be the arrival curve of the flow R.

Definition 3. Min-plus convolution
If f and g are two wide-sense functions, the min-plus convolution between f and g is

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} [f(t - s) + g(s)]$$

where inf is the infimum.

Definition 4. Service curve
Given the wide-sense function $\beta(t)$, with $\beta(0) = 0$ and considering a system S and a flow through S with the input function R and output function R^*, S is said to offer a service curve β if

$$R^* \geq R \otimes \beta$$

Theorem 1. *If a flow R is under the control of leaky bucket (σ, ρ), $R(t)$ is constrained by the arrival curve $\alpha(t) = \sigma + \rho t$. At any time t, the delay $d(t)$ satisfies*

$$d(t) \leq \sup_{t \geq 0} \{\inf\{d \geq 0; \alpha(t) \leq \beta(t + d)\}\}$$

where sup is the supremum.
The proof of the theorem can be found in [23].

4.2. Modeling mopenflow channel

We consider mOpenFlow between an SDN device and a controller having m available paths. The notations used in this section are given in Table 1. The mOpenFlow channel enables control traffic to stripe over all the m paths. The corresponding m-path model of mOpenFlow is illustrated in Fig. 4. In the figure, c_i and f_i $(i = 1, 2, \ldots, m)$ represent the available bandwidth and fixed delay of path i, respectively.

Let the cumulative OpenFlow traffic in $[0, t]$ be $R(t)$. We assume that $R(t)$ is regulated by a $\{\sigma, \rho\}$ leaky bucket, which is implemented on various routers and SDN devices. That is $R(t)$ conforms to a deterministic process, where ρ is the long-term average rate of the process, and σ is the maximum burst size (i.e., the burst tolerance factor) of $R(t)$. As the traffic is forwarded to an MPTCP scheduler, we additionally assume the control traffic is divided into multiple subflows deterministically, each of which conforms to an envelope process

$$R_i(t) = \rho_i t + \sigma_i$$

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

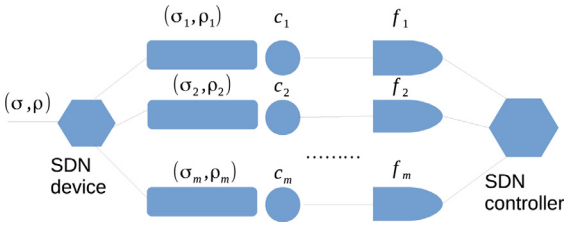

Fig. 4. Modeling the mOpenFlow channel.

with $i = 1, 2, \ldots, m$. We have another constraint

$$\sum_{i=1}^{m} R_i(t) = R(t),$$

which gives

$$\begin{cases} \sum_{i=1}^{m} \sigma_i = \sigma \\ \sum_{i=1}^{m} \rho_i = \rho \end{cases}$$

We model the bottleneck link of each path as a work conserving queueing system with a constant service rate c_i $(i = 1, 2, \ldots m)$. The queueing delay at the bottleneck link of path i is denoted as d_i $(i = 1, 2, \ldots, m)$.

4.3. Delay analysis

For the traffic along a path in mOpenFlow, the whole delay includes a propagation delay, represented by the fixed delay f_i $(i = 1, 2, \ldots, m)$ and the queueing delay d_i. We denote the whole delay along path i as D_i. Hence, D_i is the sum of the queueing delay and the fixed delay, i.e.,

$$D_i = d_i + f_i \tag{3}$$

To have a stable queue in each path, the subflow on path i should satisfy $\rho_i < c_i$, if $\sigma_i > 0$, $i = 1, 2, \ldots, m$.

In an SDWAN, the path parameters may not be constant because of the time-varying background traffic or congestion. Besides that, when an active path is broken, the replacement path k may have a different c_k or f_k. We assume that c_i and f_i $(i = 1, 2, \ldots, m)$ change on a relatively large time scale. The traffic over m paths (i.e., m subflows created by MPTCP) need to be aggregated at the buffer in the SDN controller. After the aggregation process, the contents of flows (OpenFlow messages) will be sent to a controller. Therefore, the total end-to-end delay denoted as D_{e2e}, is jointly calculated from all the delay values along the transmission paths.

Applying the findings of delay bound values in each conserving system in network calculus [24], we have the delay bound caused by the queuing on path i $(i = 1, 2, \ldots, m)$

$$d_i \le \frac{\sigma_i}{c_i - \rho_i} \tag{4}$$

Therefore, from (3) and (4) the total delay along path i is bounded by

$$D_i = \frac{\sigma_i}{c_i - \rho_i} + f_i \tag{5}$$

We then derive the upper bound for the end-to-end delay of mOpenFlow (i.e., D_{e2e}) as follows:

$$D_{e2e} = \max_{1 \le i \le m} \{D_i\} \tag{6}$$

Note that, in the case mOpenFlow's traffic is accumulated on one path (e.g., path l), the delay bound can be found by (5) with the path l's parameters. Besides, it is widely recognized that MPTCP performs badly when the paths are imbalanced. To capture that behavior in mOpenFlow, we investigate the variation in end-to-end delay values of available paths. The upper bound of variation V_m can be derived:

$$V_m = \max_{1 \le i \ne j \le m} \left| D_i - D_j \right| \tag{7}$$

Hence

$$V_m = \max_{1 \le k \le m} \frac{\sigma_k}{c_k - \rho_k} + \max_{1 \le i \ne j \le m} \left| f_i - f_j \right| \tag{8}$$

If path parameters are available, then the SDN device can estimate the end-to-end latency bounds. In an SDWAN with predetermined latency requirements, the SDN device can be configured to remove unsuitable paths for control traffic.

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

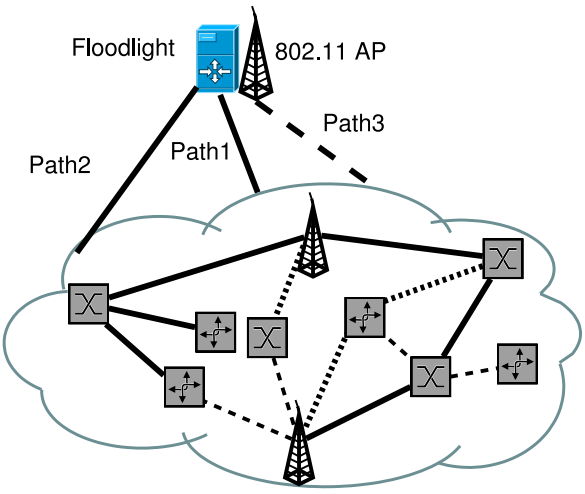

Fig. 5. SDWAN with three available paths for OpenFlow channel.

5. Evaluation

5.1. Environment settings

We implement a prototype of mOpenFlow in an emulated SDWAN, which includes the high performance Floodlight controller [25] and a number of SDN devices. The devices are emulated using the well-known tool, namely *cbench* [8]. We position the *cbench* and the controller in two different computers (8x2.5Ghz 8GB RAM) to minimize possible interference on the control channel [8]. The computers use Ubuntu 13.10 with the version 0.88 of mptcp-kernel. Although we have chosen to use the Floodlight controller in this work, mOpenFlow can work with other OpenFlow controllers as well. The configuration of Floodlight has been inherited from[2].

As depicted in Fig. 5, three paths are available (i.e., path1, path2, path3) for the control channel in the SDWAN. Path1 and path2 are Ethernet cables, and path3 is a Wi-Fi link. We use the linux tc to shape different bandwidth values the path1 and path2 for two different scenarios: (i) one SDN device and (ii) ten devices. The tool is also used to add delay to path2 to make a different path configuration. We benchmark the paths' performances (i.e., bandwidth and round trip time (RTT)) using iperf and ping. The benchmarking results (bandwidth, RTT average/deviation) in the one device scenario as follows: path1 (10 Mbps, 1.154/0.084 ms), path2 (10 Mbps, 15.391/3.065 ms), and path3 (13.2Mbps, 5.171/0.841 ms). In the ten-device scenario, the RTT values are kept but the bandwidth of two wired paths is 100 Mbps.

In the evaluation, cbench generates a test series, through each of which, a device sends a configurable number of *Packet In* messages to the controller. The device waits for appropriate response messages (e.g., *Flow Mod* or *Packet Out*), and reports the performance values. Cbench supports two operation modes: latency and throughput. In the latency mode, each device maintains exactly one new flow request, waiting for a success response, before soliciting the next request; in the throughput mode, each device maintains as many requests as the TCP buffer on the cbench machine allows. We use both these modes in different experiments to assess the scalability and the robustness of mOpenFlow.

5.2. Scalability evaluation

In this evaluation, we use the throughput mode of cbench. We compare the performance of mOpenFlow (mOF) to that of the standard channel (sOF) in the two scenarios. In each scenario, first we let cbench uses the sOF and challenge the controller via each single path. Then mOF is used with two paths (i.e., path1 and path3), and mOF via all the three paths. In each cbench run, cbench is in the throughput mode in a period of ten seconds. Under each setting in the scenarios, a cbench run is repeated ten times. The results, including the values of average throughput and standard deviation, are collected and plotted in Fig. 6, Fig. 7 for the one-device and ten-device scenario, respectively.

Along the *x*-axes, sOF*i* stands for the standard OpenFlow channel over path *i*, and mOF-*n* is for mOpenFlow via *n* paths. The *y*-axes show the number of successful flows handled on the channel. As shown in the figures, the sOF1's achievable throughput values are higher than those of sOF2. This is because the path2's RTT is longer than the path1's. On the other hand, the throughput values of path3 with sOF in the two scenarios are similar. That confirms the correctness of environment settings since there is no change in the Wi-Fi link during the evaluation. We also observe that, in all the cases, the

[2] https://floodlight.atlassian.net/wiki/pages/viewpage.action?pageId=1343657 (Accessed: 2016 July)

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

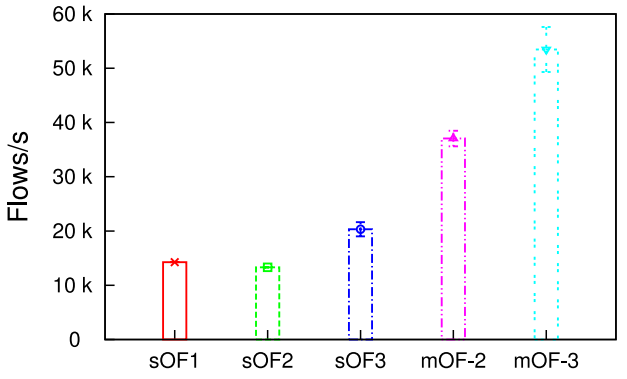

Fig. 6. Scalability evaluation of sOF and mOF in one-device scenario.

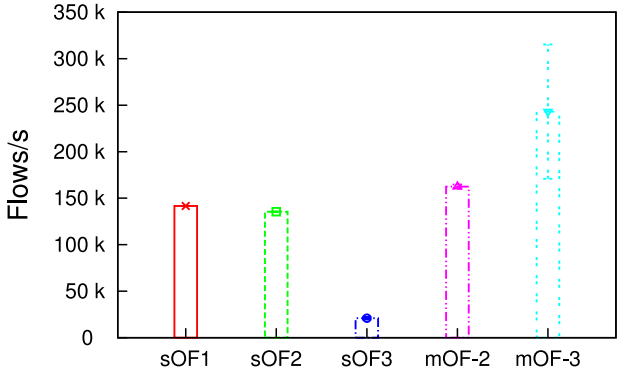

Fig. 7. Scalability evaluation of sOF and mOF in ten-device scenario.

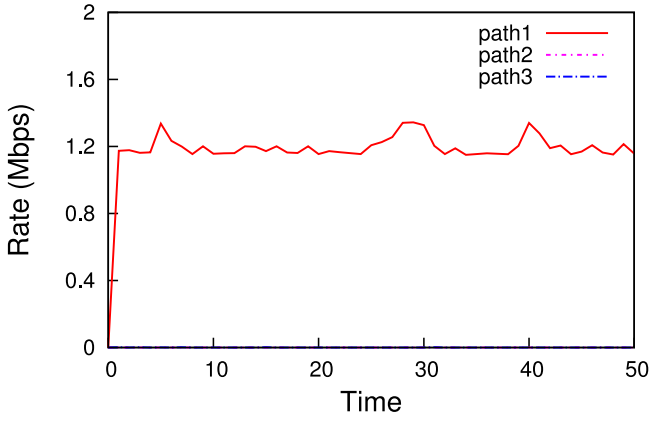

Fig. 8. mOpenFlow on only path1 in one-device scenario.

throughput of standard OpenFlow is smaller than that of mOpenFlow. Moreover, the achievable throughput of mOpenFlow increases along with the number of paths. That confirms the scalability of mOpenFlow.

5.3. Robustness evaluation

The robustness evaluation includes two experiments. In the first one, we evaluate the resilience against path failure of mOpenFlow in the one-device scenario. In the second one, we evaluate the robustness via the capacity of OpenFlow channels when they sustain loss on the paths.

The latency mode of cbench is used in the first experiment. The SDN device transmits messages to the controller one by one. During the transmission process, the mOpenFlow channel randomly experiences failures. We use the tool bwm-ng to track the variations of transmitting rate on the three paths. The tracking results are shown in Figs. 8, 9, and 10 for the

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

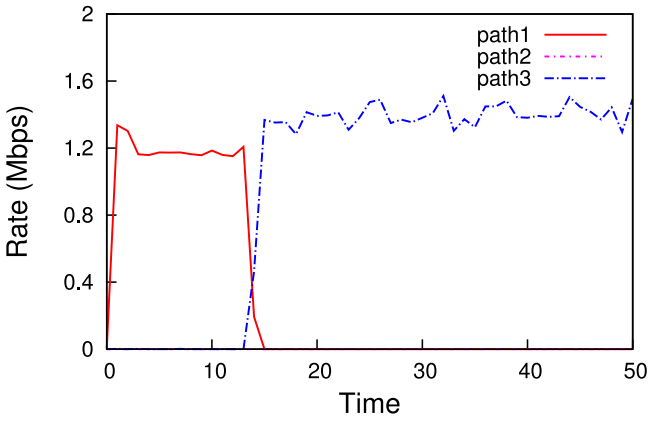

Fig. 9. mOpenFlow with path1 failure.

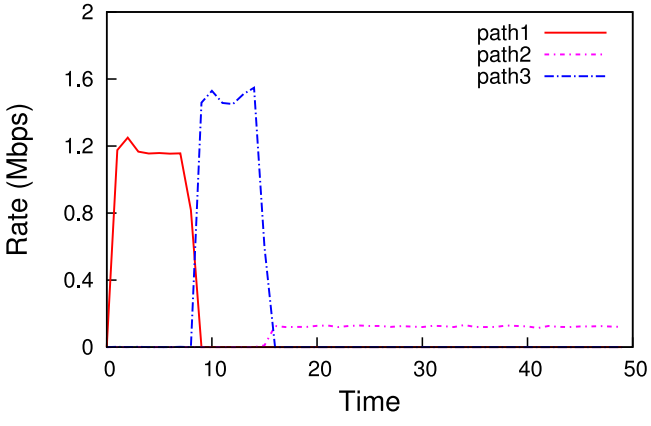

Fig. 10. mOpenFlow with path1 and path3 failures.

cases of zero, one, and two failed paths, respectively. Fig. 8 indicates that mOpenFlow uses the lowest latency path (i.e., path1) for transferring OpenFlow messages. When there is an error on path1 that terminates the transfer, the OpenFlow messages are switches to path3 (i.e., wireless path) as in Fig. 10. In the cases of two failed paths in Fig. 10, at the beginning of experiment, the OpenFlow messages are also on path1. When path1 gets failed, the traffic is automatically switched to path3. Finally, a similar switchover process happens to path2 when path3 experiences failure. Therefore, we can conclude the mOpenFlow channel is resilient against path failures.

The second experiment is a repeat of the scalability evaluation in the ten-device scenario. However, the paths are configured with loss probability values. The results of throughput and the deviation values over the two OpenFlow channels are shown in Fig. 11. In the figure, the *x*-axis shows the loss rate, which is varied from 0 to 10%, and the y-axis shows the throughput values. Generally, the values decrease when the loss rate increases. However, mOpenFlow is more loss-resilient than the standard OpenFlow.

5.4. Numerical analysis results

We first verify the accuracy of our analysis by comparing experimental and analytical values. In our experiments, we employ leaky buckets for the three paths with {1.4Mbps, 150Kbps} for path1, path2, and {1.0 Mbps, 140 } for path3. In order to capture the stability of paths (i.e., constant path capability c_i) we set the duration of cbench's latency mode is one second. Note that, cbench has no regulated leaky bucket during traffic generation; we use the latency mode in this evaluation. We found that depending on the initialization path (i.e., on which the first SYN is sent), mOF has different behavior. In each case of the initialization, we run 100 experiments and calculate the maximum delay from the trace data. The experimental delay values are plotted in Fig. 12 in a comparison to the analytical bound. Obviously, in this scenario the delay bound is relied on path2's parameters. As shown in the figure, we can see that when the mOF's initializations are on shorter latency paths (i.e., path1 and path3), the OpenFlow traffic is conveyed on one path. The latency is far lower than the bound. However, in the case of path2, the OpenFlow traffic has been spread over more than one path, hence the multipath analysis model could

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

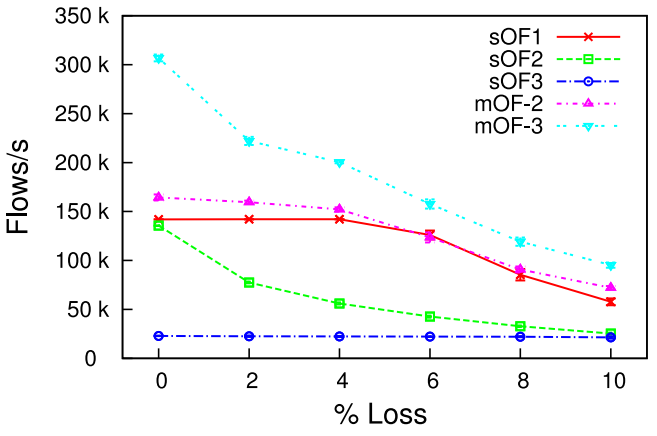

Fig. 11. Robustness evaluation under loss environment.

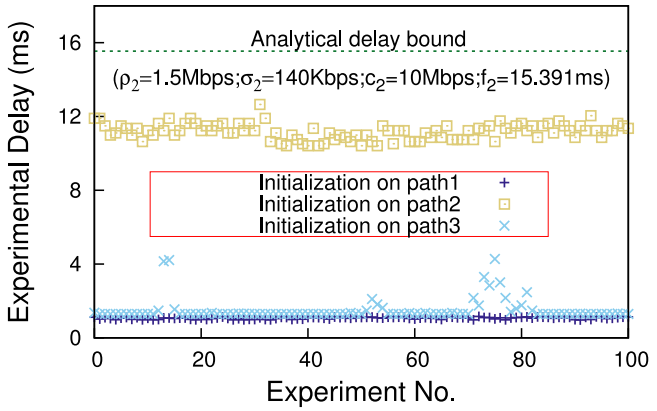

Fig. 12. Comparing the numerical analysis and experimental data.

be compared more accurately. In the such case, the experimental delay becomes much higher, however all the values are below the analytical bound. That indicates the correctness of our analysis.

Following in this section, we give numerical examples in order to demonstrate the applications of the proposed model. We adopt the benchmarking values of RTT (i.e., the average values) in the previous section as baseline parameters. We then use the delay analysis in Section 4 to investigate the bounds of end-to-end delay in the two cases of mOpenFlow-2 with path1 and path2 (i.e., Ethernet cables), mOpenFlow-2 with path1 and path3 (i.e., wire and wireless). We consider the cases of mOpenFlow-2 concurrently using both two paths.

In the case of mOpenFlow-2 with path1 and path2, the maximum burst size and the sustained rate are the same in both the paths ($\{\rho_i, \sigma_i\} = \{1.5\text{Mbps}, 140\text{Kbps}\}$, with $(i = 1, 2)$, $f_1 = 1.154\text{ms}$, $f_2 = 15.391\text{ms}$). The relationship between the available bandwidth on two paths and the bounds of end-to-end latency on mOpenFlow-2, which is derived from (6), is plotted in Fig. 13. In the figure, the red curve indicates the bound of end-to-end latency. In the other case, the sustained rate and the burst size are different on the two paths ($\{\rho_1, \sigma_1\} = \{1.5\text{Mbps}, 140\text{Kbps}\}$, $f_1 = 1.154\text{ms}$, and $\{\rho_3, \sigma_3\} = \{1.0\text{Mbps}, 140\text{Kbps}\}$, $f_3 = 5.171\text{ms}$). The similar analysis are adopted to reveals the delay bound of this case in Fig. 14. From Figs. 13 and 14, we can observe that when the available bandwidth increases, the bounds of end-to-end delay on mOpenFlow-2 channel decrease. This, in effect, means that the increase in bandwidth is expected to allocate for the control traffic to get the lower bounds.

6. Conclusion

In a future SDWAN, to keep the OpenFlow traffic and protocol working efficiently, the control channel is required to be robust, scalable, and evolvable. In this paper, we propose the mOpenFlow channel, which fulfills these requirements. The proposed channel uses the standard MPTCP in conveying the OpenFlow traffic; hence, it is not only compatible with SDWAN but also adoptable within the OpenFlow standard. The multipath communication capability increases both the robustness and scalability of OpenFlow channel. We evaluate the mOpenFlow (mOF) in the emulated SDWAN relating to the standard channel (sOF). The results show that mOF is not only more loss-resilient than sOF, but also resilient against path failures. The

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

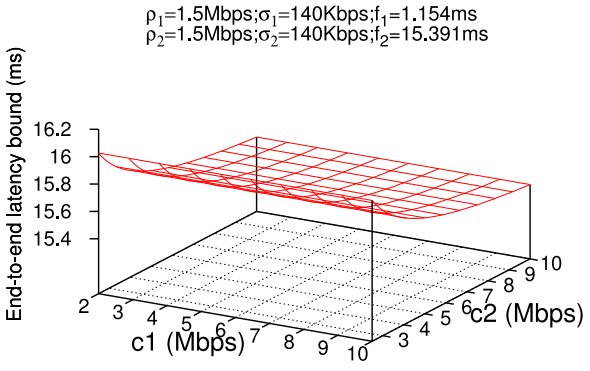

Fig. 13. End-to-end delay analysis of mOpenFlow-2 using path1 and path 2.

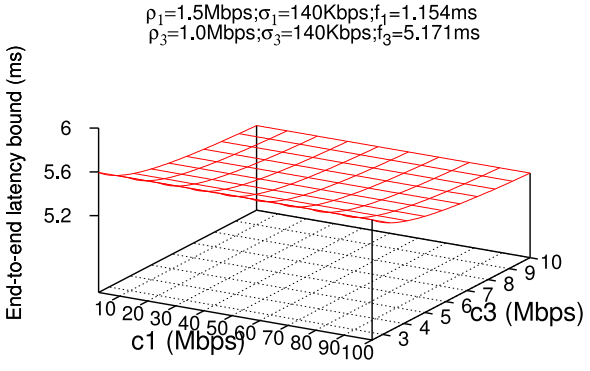

Fig. 14. End-to-end delay analysis of mOpenFlow-2 using path1 and path3.

achievable throughput of mOF is significantly higher than that of sOF. On the other hand, we propose a network calculus-based framework to model mOpenFlow for performance analysis. The numerical analysis, relying on the network-calculus framework, proves to be a fast and convenient way for analyzing the end-to-end delay on mOF.

Acknowledgement

This research was conducted under a contract of R&D for radio resource enhancement, organized by the Ministry of Internal Affairs and Communications, Japan.

References

[1] Jin X, Li LE, Vanbever L, Rexford J. SoftCell: scalable and flexible cellular core network architecture. In: Proc. ACM CoNEXT; 2013. p. 163–74.
[2] Yap K, Kobayashi M, Sherwood R, Huang T-Y, Chan M, Handigol N, et al. Openroads: empowering research in mobile networks. SIGCOMM Comput Commun Rev 2010;40(1):125–6.
[3] Moradi M, Li LE, Mao ZM. SoftMoW: Recursive and Reconfigurable Cellular WAN Architecture. In: Proc.ACM CoNEXT; 2014.
[4] Bernardos C, De La Oliva A, Serrano P, Banchs A, Contreras L, Jin H, et al. An architecture for software defined wireless networking. IEEE Wireless Commun 2014;21(3):52–61.
[5] Open Networking Foundation. OpenFlow switch specification 1.4. 2013.
[6] Nguyen K, Ishizu K, Murakami H, Kojima F, Yano H. A scalable and robust OpenFlow channel for software defined wireless access networks. In: Proc. IEEE VTC-Fall; 2015.
[7] Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, et al. Onix: a distributed control platform for large-scale production networks. In: Proc. USENIX OSDI; 2010. p. 1–6.
[8] Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. In: Proc. Hot-ICE; 2012. p. 10–16.
[9] Beheshti N, Zhang Y. Fast failover for control traffic in software-defined networks. In: Proc. IEEE GLOBECOM'12; 2012. p. 2689–94.
[10] Desai M, Nandagopal T. Coping with link failures in centralized control plane architectures. In: Proc. COMSNETS; 2010. p. 79–88.
[11] Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, et al. B4: experience with a globally-deployed software defined wan. In: Proc. ACM SIGCOMM; 2013. p. 3–14.
[12] Nguyen K, Yamada S. An experimental feasibility study on applying SDN technology to disaster-resilient wide area networks. Ann Telecommun 2016:1–9. doi:10.1007/s12243-016-0502-2.
[13] Jarschel M, Oechsner S, Schlosser D, Pries R, Goll S, Tran-Gia P. Modeling and performance evaluation of an OpenFlow architecture. In: Proc. IEEE ITC; 2011. p. 1–7.
[14] Azodolmolky S, Nejabati R, Pazouki M, Wieder P, Yahyapour R, Simeonidou D. An analytical model for software defined networking: a network calculus-based approach. In: IEEE GLOBECOM; 2013. p. 1397–402.
[15] Cholda P, Jajszczyk A. Recovery and its quality in multilayer networks. J Lightwave Technol 2010;28(4):372–89.

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

[16] Vulimiri A, Godfrey PB, Mittal R, Sherry J, Ratnasamy S, Shenker S. Low latency via redundancy. In: Proc. ACM CoNEXT; 2013. p. 283–94.
[17] IEEE 802.1AX. http://standards.ieee.org/findstds/standard/802.1AX-2008.html; (Accessed: 2016 July).
[18] Touch J., Perlman R. Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement. RFC 5556 (Informational); 2009.
[19] Allan D, Ashwood-Smith P, Bragg N, Farkas J, Fedyk D, Ouellete M, et al. Shortest path bridging: Efficient control of larger ethernet networks. Commun Mag 2010;48(10):128–35.
[20] Hopps C. Analysis of an equal-cost multi-path algorithm. 2000.
[21] Raiciu C, Paasch C, Barre S, Ford A, Honda M, Duchene F, et al. How hard can it be? Designing and implementing a deployable multipath TCP. In: Proc. USENIX NSDI; 2012.
[22] Ford A., Raiciu C., Handley M., Bonaventure O. TCP extensions for multipath operation with multiple addresses. RFC 6824; 2013.
[23] Le Boudec J-Y, Thiran P. Network calculus: a theory of deterministic queuing systems for the internet. Berlin, Heidelberg: Springer-Verlag; 2001.
[24] Chang C-S. Stability, queue length, and delay of deterministic and stochastic queueing networks. IEEE Tran Autom Control 1994;39(5):913–31.
[25] Floodlight OpenFlow Controller. http://www.projectfloodlight.org/floodlight/; (Accessed: 2016 July).

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014

Kien Nguyen received the Ph.D. degree in Informatics from The Graduate University for Advanced Studies (SOKENDAI), Japan in 2012. He is currently a researcher at National Institute of Information Communications Technology (NICT), Japan. He is also a senior member of IEEE. His current researches focus on evolvable networking technologies for the 5G mobile wireless networks such as MPTCP, SDN.

Kentaro Ishizu received Ph.D from Kyushu University in 2005 in computer science. Since then, he has been working with NICT of Japan on Heterogeneous wireless networks, cognitive radio networks, TV white space communications. He also has been involved with international standardizations including IEEE802.21 and IEEE1900.4. His current interests include spectrum sharing for future generation wireless networking.

Fumihide Kojima received the D. E. degrees in electrical communications engineering from Osaka University, Japan, in 1999. Currently, he is a director of Wireless Systems Laboratory of National Institute of Information and Communications Technology (NICT). His research includes intelligent MAC protocol for radio communication systems including smart utility networks.

Please cite this article as: K. Nguyen et al., An evolvable, scalable, and resilient control channel for software defined wireless access networks, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.09.014