Priority–based Scheduling Policy for OpenFlow Control Plane

Piyawad Kasabai[1], Karim Djemame[2], and Somnuk Puangpronpitag[1]
[1]Mahasarakham University
Thailand
[e-mail: {piyawad.k, somnuk.p}@msu.ac.th]
[2]School of Computing, University of Leeds
Leeds - UK
[e-mail: k.djmame@leeds.ac.uk]
*Corresponding author: Somnuk Puangpronpitag

Received January 5, 2018; revised June 30, 2018; revised August 27, 2018; accepted September 26, 2018; published February 28, 2019

Abstract

Software Defined Networking (SDN) is a new network paradigm, allowing administrators to manage networks through central controllers by separating control plane from data plane. So, one or more controllers must locate outside switches. However, this separation may cause delay problems between controllers and switches. In this paper, we therefore propose a Priority-based Scheduling policy for OpenFlow (PSO) to reduce the delay of some significant traffic. Our PSO is based on packet prioritization mechanisms in both OpenFlow switches and controllers. In addition, we have prototyped and experimented on PSO using a network simulator (ns-3). From the experimental results, PSO has demonstrated low delay for targeted traffic in the out-of-brand control network. The targeted traffic can acquire forwarding rules with lower delay under network congestion in control links (with normalized load > 0.8), comparing to traditional OpenFlow. Furthermore, PSO is helpful in the in-band control network to prioritize OpenFlow messages over data packets.

Keywords: Software Defined Networking, OpenFlow, Switch-Controller Delay

1. Introduction

Traditional IP networks are complex and very hard to manage [1]. Classical switching and routing devices on this traditional IP networks are inflexible to optimize. These devices integrate both data plane and control plane on the same hardware devices. So, the Software Defined Networking (SDN) architecture [2] has been proposed to separate and operate between data plane and control plane. In general, the data plane locates in switch hardware, whereas the control plane is software running on one or more servers, called 'controller'. The data plane provides the simplest function of switches, i.e., forwarding packet according to a set of rules. The rules in the switch are managed by software at the controller. This SDN architecture is designed to provide various perspectives (such as manageability and flexibility) in the programmable networks that traditional network suffers from. So far, there have been several SDN-based solutions, such as SoftRouter [3], ForCES [4], and OpenFlow [5].

OpenFlow has been widely deployed in various network products, and attracts several network industries [6]. The OpenFlow protocol defines control messages to handle a switch. The control messages may be sent on a separated network from the data traffic (called out-of-band control network), or may be sent on a shared network infrastructure with the data traffic (called in-band control network). Since SDN networks grow in scale and complexity, the control traffic may suffer from delay, resulting in network inefficiency [7]. Several solutions ([7]–[11]) have also been proposed to reduce the delay of the control traffic. However, some solutions support only a specific communication between the control and data planes (in-band or out-of-band control networks) [8], [9]. Some solution [7] has proposed an initial design to fix this problem, but with a rather high overhead for traffic tagging.

Hence, this paper proposes a Priority-based Scheduling policy for OpenFlow (PSO) to fix the aforementioned problems. The purpose of PSO design is twofold: (1) to overcome the bandwidth competition between data traffic and control traffic for the in-band control network, (2) to provide high-priority packet-in messages, based on packet contents (such as real-time services) for both in-band and out-of-band control network.

The remainders of this paper are organized as follows. Section 2 gives background reviews of SDN and OpenFlow switches. The motivation of this research is explained in Section 3, and our PSO design is described in Section 4. The evaluation of PSO is discussed in Section 5, and compared to related work in Section 6. Finally, Section 7 concludes this work.

2. Background

2.1 Software-Defined Networking (SDN)

In general, network devices are vender-dependent and closed systems. They are inflexible to optimize for different levels of services. Several studies [3]–[5], [12] have therefore proposed to solve the problem by implementing data handling rules as software rather than embedding them into hardware. Software Defined Networking (SDN) is one of the most well-known solutions. It provides a new approach for network administrators to manage network functionality and provision. SDN focuses on the role of software in running networks through an abstraction of the data plane, and separating it from the control plane. This separation allows faster innovation cycles at both planes. SDN architecture consists of multiple planes, including Forwarding Plane (FP), Operational Plane (OP), Control Plane (CP), Management

Plane (MP), and Application Plane (AP). Further details of SDN layers and architecture can be found in RFC 7426 [13].

2.2 OpenFlow

OpenFlow is a well-known SDN implementation. An OpenFlow switch [5], [6], [14] forwards data packets according to a set of rules in flow tables. These rules are managed by a software-based controller at the control plane outside the switch. According to the OpenFlow specification [14], there are three main components of the OpenFlow switch: *a secure channel*, *flow tables* and *an OpenFlow protocol*. The *secure channel* (also known as open channel) is a software API to connect with the controller, allowing commands and packets to communicate between the controller and the switch. The *flow tables* are built inside the switch hardware using Ternary Content Addressable Memory (TCAM). They contain a list of flow entries, defining rules for forwarding/dropping/modifying packets. Each flow entry consists of *match-field*, *counter* and *instruction*. For each incoming packet, the packet header is compared to the match-field of each entry. If the packet header is matched with any flow entry, the switch then takes the action of the instructions in that flow entry. If the header of the packet is not matched with any flow entry, this event is called *table-miss*. The packet is then encapsulated into an OpenFlow packet-in message. After that, the switch sends the packet-in message to the controller via an SDN interface to request an action or a new flow entry that will be stored in the flow tables. The controller responds by sending an OpenFlow packet-out message, and maybe an OpenFlow flow-mod message back to define a proper flow-entry for the packet at the switch. Since the controller is software-based, so it can be dynamically programed to provide manageability.

The OpenFlow protocol provides a standard for communication between controllers and switches by defining several types of control messages, such as symmetric messages, switch configuration messages, asynchronous messages, and controller command messages. For example, OFPT_PACKET_IN is one of asynchronous messages, describing packet-in messages. The details of OpenFlow protocol can be found in [14].

2.3 In-band Control Network vs. Out-of-band Control Network

To communicate between control and data planes, there are two alternatives, namely in-band control network and out-of-band control network. For the out-of-band control network, control traffic (the OpenFlow control message) is sent on a separate network from data traffic. Yet, for the in-band control network, the control and data traffic share the same network link. From the literature, the out-of-band control network has been focused by several studies [15]. It is also used by B4 (Google Software Defined WAN) [16]. Its advantages are as follows: (1) high security can be provided for control messages; (2) high availability can be provided even if there are failures in some network devices. However, this out-of-band control network is expensive to build due to the separation of network link. Sharma et al. [9], [10] have also suggested that the in-band control network is suitable for all types of topologies.

2.4 Centralized Controller vs. Distributed Controller

Centralized and distributed controllers are two alternatives for SDN controller placement. For the distributed controllers, inter-connection links among controllers [17] must be built. White and Zandy [18] have suggested that the distributed controllers are rather complex, and require heavy configuration to design, deploy, and manage. On the other hand, the centralized controller is much simpler. So, the centralized controller is more widely deployed comparing

to the distributed controllers. However, SDN can grow in scale. The number of the switches under the same centralized controller could then be increased. This inevitably causes the network congestion problem [7]. This work will focus on the centralized controller.

2.5 Differentiated Services Code Point (DSCP) of DS field

DiffServ [19] is a traffic control architecture, relying on the 8-bit Differentiated Services (DS) field (in place of the outdated Type of Service (ToS) field [20]) in the IP header. DS field consists of the first six bits for the Differentiated Services Code Point (DSCP) and the other two bits for Explicit Congestion Notification (ECN). Due to its six-bit length, DSCP can support up to 64 different classes of traffic. DiffServ routers then decide on per-hop basis how to forward packets based on their class. First three bits of DSCP indicates IP precedence. These bits are called Class Selector (CS), prioritizing traffic types by classes (CS0 – CS7, lowest to highest priorities respectively). In our design, DSCP is deployed to mark the traffic priorities. The OpenFlow switch is extended by implementing a queuing mechanism to classify traffic according to the DSCP values. Some classes may be reserved for the OpenFlow control traffic (further described in Section 4.2).

3. Problems and Motivation

Most of the manageability in SDN relates to decoupling of the control and data planes. In particular, the first packet of a new flow is sent by a switch to the controller to acquire a forwarding rule. This may increase network load, and make the control plane a potential bottleneck [21]. In addition, since the flow tables of switches are configured in real time by an external device, there is also the extra delay, introduced by the flow setup process.

In the centralized controller environment, there may be several switches under the same controller, resulting in the bandwidth competition among OpenFlow control messages in the out-of-band control network. This competition could finally increase the transmission delay of data traffic. Without considering different data traffic types and prioritizing their control traffic properly, some delay sensitive services may finally fail. Network inefficiencies may then occur in SDN. Hsiao et al. [22] has previously pointed out that SDN suffers from the transmission delay, and this transmission delay is a significant issue of transmission quality for network operators.

Furthermore, the situation in the in-band control network would be even worse than the out-of-band control network. There is an extra competition between OpenFlow control messages and data packets. Without having higher priority, the OpenFlow control messages may be dropped. This should cause the failure of data traffic forwarding at the end.

Hence, this paper proposes a Priority-based Scheduling Policy for OpenFlow (PSO). The design of PSO is to give a higher priority for control traffic in the in-band control network. Furthermore, PSO provides different priorities for OpenFlow control messages, based on contents/services (data traffic types) for both in-band and out-of-band control networks.

4. Design

4.1 PSO Modules

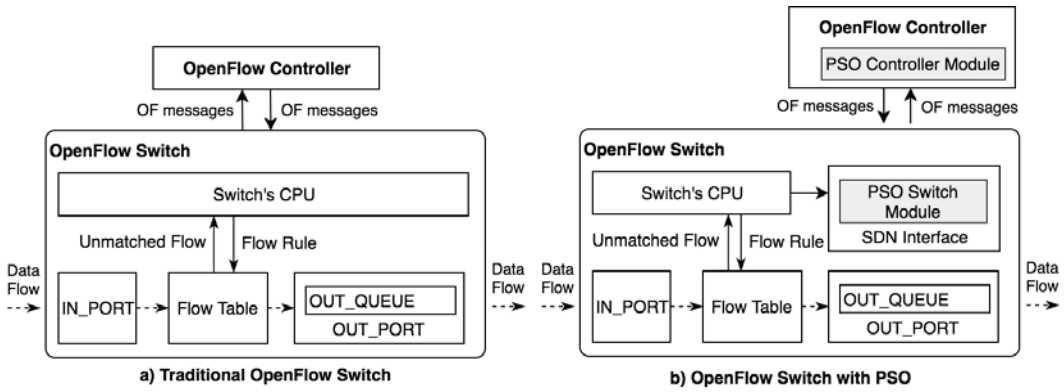

Fig. 1. Traditional OpenFlow switch vs. OpenFlow switch with PSO

Fig. 1a shows an architecture of traditional OpenFlow switches. Generally, an OpenFlow switch looks up all incoming packets from an IN_PORT queue to match with its flow tables. The incoming packets are then served at an OUT_PORT queue, by taking actions according to the rule in the matched flow entry. In general, queue management is FIFO (First In First Out) with a drop-tailed algorithm. All packets are treated equally. So far, there has been no special queuing functionality, proposed in OpenFlow protocol [9]. According to our PSO design (illustrated in **Fig. 1b**), PSO modules are embedded into both OpenFlow switches (PSO switch module) and controllers (PSO controller module) respectively. These modules provide a special queuing mechanism to automatically prioritize OpenFlow messages and data traffic.

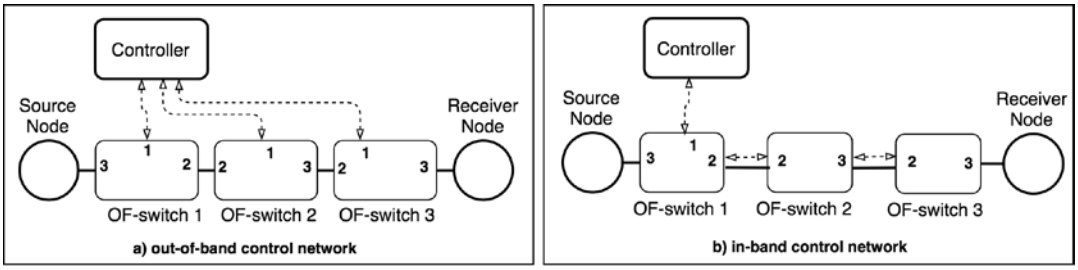

Fig. 2. SDN interfaces

Instead of using FIFO, a PSO switch module provides a special queue management on SDN interfaces of the OpenFlow switch. For the out-of-band control network, the SDN interface (or SDN port) of an OpenFlow switch is a specific port of that switch, directly connecting to the OpenFlow controller [23]. As shown in **Fig. 2a**, port-1 of the OF-switch 1, port-1 of the OF-switch 2, and port-1 of OF-switch 3 are SDN interfaces. For the in-band control network, some specific ports on each switch are deployed to pass OpenFlow control messages to the controller. We also call them SDN interfaces. Some of these SDN interfaces

may directly connect to the controller, for example, port-1 of the OF-switch 1 (as shown in **Fig. 2b**). Otherwise, some SDN interfaces on the in-band control network may indirectly connect to the controller via the other switches. For example, port-2 of the OF-switch 1, port-2 and 3 of the OF-switch 2, and port-2 of OF-switch 3 are SDN interfaces (as shown in **Fig. 2b**). As previously mentioned, these SDN interfaces in the in-band network could suffer some delay due to the competition between OpenFlow control messages and data packets transmitting over the same interface. Our PSO switch module is designed to mitigate this problem by providing special queue management on these interfaces to prioritize OpenFlow messages over data packets. For out-of-brand control network, PSO can provide ability to assist some services (such as real-time applications or the services of high-priority users) to be processed with lower delay.

At the controller, a PSO controller module will provide special queue management for all interfaces to prioritize OpenFlow messages.

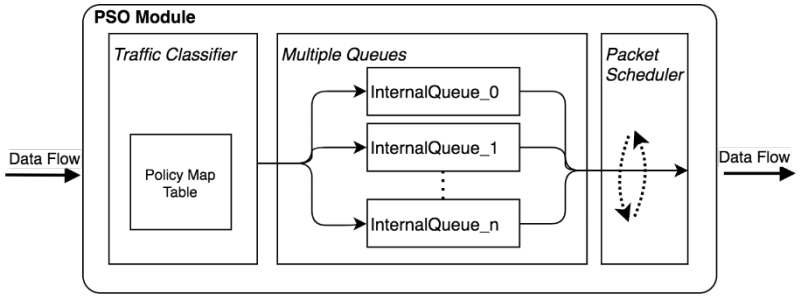

Fig. 3. The components of PSO modules

Both PSO switch and controller modules consist of a *traffic classifier*, *multiple queues*, and a *packet scheduler* (as illustrated in **Fig. 3**). The *traffic classifier* differentiates packets by using a Policy Map Table (PMT). This PMT is built inside TCAM like flow tables. The *multiple queues* are internal queues for traffic with different priorities. The *packet scheduler* is a packet prioritization scheduling mechanism. The details of the prioritization will be further discussed in section 4.3. The PSO controller module is designed to follow the prioritization set by the PSO switch module. It has a queuing mechanism corresponding to the PSO switch module.

4.2 Traffic Classifier

For any traffic arriving at an SDN interface, a *traffic classifier* will differentiate traffic according to a set of predefined rules in a *Policy Map Table (PMT).* The rules must be set by a network administrator at the controller. Otherwise, the traffic will be treated equally. The PMT will then be copied to all OpenFlow switches in the network using OFPT_SET_CONFIG, which is an OpenFlow control message for switch configuration. Each rule of PMT contains *traffic type*, *match-fields*, and an *action* (as shown in **Fig. 4**). The *traffic classifier* will match the arriving traffic with *traffic type* and *match fields*, then follows the *action* of the matched record.

traffic type	*match-fields*	*action*

Fig. 4. Policy Map Table (PMT)

Table 1. *Traffic types*

traffic type no.	traffic type description
1	OpenFlow configuration messages, and OpenFlow symmetric messages
2	OpenFlow packet-in message, OpenFlow packet-out messages, and other OpenFlow control command messages
3	Other OpenFlow messages
4	data packets

traffic type is a field to specify different types of traffic, as shown in **Table 1**. In this paper, we predefine four *traffic types* that should be treated with different priorities accordingly. The first type includes *OpenFlow configuration messages* (e.g., OFPT_SET_CONFIG) and *OpenFlow symmetric messages* (e.g., OFPT_ECHO_REQUEST, OFPT_ECHO_REPLY). The second type includes OFPT_PACKET_IN, OFPT_PACKET_OUT messages and *other control command messages*. The third type includes other OpenFlow control messages, such as OFPT_TABLE_STATUS. Finally, the forth type includes data packets. In the in-band control network, the data packets may share the same link with OpenFlow control messages. So, we give OpenFlow control messages higher priorities than data packets. Among the OpenFlow control messages, we also give three different priorities as shown in the **Table 1**. OpenFlow configuration messages are given the highest priority to ensure that any configurations by network administrators work out on time. *Packet-in messages* (OFPT_PACKET_IN), *packet-out messages* (OFPT_PACKET_OUT) and *other control command messages* are given a higher priority than other OpenFlow control messages (such as OFPT_TABLE_STATUS) since they carry important instructions between the controller and the OpenFlow switches. The details of the types of OpenFlow control messages can be found from [14]. These predefined traffic types and priorities are also flexible, and may be specified differently by network administrators for different organizations.

match-fields are the match-fields of flow tables, which details are given in the OpenFlow specification [14]. They contain several header fields to match against the header of data packets. The match fields can help specify application services (for example, protocol=TCP port=80 is specified “http” service).

action contains an action defining how the traffic should be treated by the *packet scheduler*. *action* can be setting DSCP values, or setting output queue ID. The following examples show how PMT could be set to prioritize different traffic according to traffic types and services in both out-of-band and in-band control networks.

Example 1: a PMT for an out-of-band control network to give a priority for a specific service is defined as follows.

```
Rule #1: traffic_type=1, action=queue_id:0
Rule #2: traffic_type=2, ip_proto=17, udp_dst=20000, action=queue_id:1
Rule #3: traffic_type=2, action=queue_id:2
Rule #4: traffic_type=3, action=queue_id:3
```

From the Example 1, a PMT contains four rules as follows. *Rule #1* is to set queue_id=0 (the highest priority queue) for all *OpenFlow configuration and symmetric messages*. This is

to give the highest priority to configuration commands from network administrators. *Rule #2* is to set queue_id=1 (the second highest priority queue) for *OpenFlow packet-in/packet-out messages* of real-time services (UDP destination port=20000). *Rule #3* is to set queue_id=2 (the third highest priority queue) for *OpenFlow packet-in/packet-out messages* of other services. *Rule #4* is to put all other OpenFlow messages into queue_id=3 (the lowest priority queue). In summary, example-1 differentiates traffic by looking at its traffic types, and its services (using *match-fields*), and put the different priority traffic into different queues. The packet scheduler of this case manages four internal queues using Priority Queue or *min rate* [14].

This PMT helps give the higher priority to OpenFlow message to acquire forwarding rule of a specific service in comparison to other service, thus reducing the delay of acquiring forwarding rule.

Example 2: a PMT for an in-band control network to give a priority to OpenFlow control messages in competing with data packets is defined as follows.

Rule #1: traffic_type=1, action=dscp:CS7
Rule #2: traffic_type=2, action=dscp:CS6
Rule #3: traffic_type=3, action=dscp:CS5
Rule #4: traffic_type=4, action=dscp:copy

In this example, there are both OpenFlow control messages and data packets, competing on the same connection due to an in-band control network. *Rule #1* will set DSCP header of packets to CS7 for all *OpenFlow configuration and symmetric messages*. This is to give the highest priority to configuration commands by network administrators. *Rule #2* gives the second priority to *OpenFlow packet-in/packet-out messages*, and *other OpenFlow control command messages*, by setting their DSCP headers to CS6. *Rule#3* gives the third priority to *other OpenFlow messages*, by setting their DSCP headers to CS5. Finally, *Rule#4* gives the lowest priority to data packets, and keeps their DSCP values at the packet headers as the old values. In this case, DSCP of the data packets may be previously set to give different priorities. These DSCP values of data packets should be defined less than CS5. In the other case, DSCP of the data packets may not be set; thus, all data packets are treated equally. These DSCP values of OpenFlow control messages and data packets will be then considered by a packet scheduler to schedule the traffic according to their priorities (such as using WFQ). This PMT helps give the higher priority to OpenFlow message in comparison to data packets, thus reducing the delay of OpenFlow messages. It also helps give different priorities to different OpenFlow message types according to their significances.

Example 3: a PMT for an in-band control network to give a priority to OpenFlow control messages and a specific service is defined as follows.

Rule #1: traffic_type=1, action=dscp:CS7
Rule #2: traffic_type=2, ip_proto=17, udp_dst=20000, action=dscp:CS6
Rule #3: traffic_type=2, action=dscp:CS5
Rule #4: traffic_type=3, action=dscp:CS4
Rule #5: traffic_type=4, action=dscp:copy

Example-3 demonstrates how PMT prioritizes traffic according to traffic types and services in an in-band control network. From the example, a PMT contains five rules in an in-band control network. *Rule #1* is the same as the one given in the example-2. It gives the highest priority to configuration commands by network administrators by setting DSCP header of packets to CS7 for OpenFlow configuration and symmetric messages. *Rule #2* gives the second priority to *OpenFlow packet-in/packet-out messages* of real-time services (UDP port 20000) by setting their DSCP header to CS6. *Rule #3* gives the third priority to *OpenFlow packet-in/packet-out message* of other services by setting their DSCP header to CS5. *Rule #4* gives a lower priority (DSCP=CS4) than *Rule #3* to *other OpenFlow messages.* The last rule (Rule #5) gives the lowest priority to data packets, and set their DSCP headers equal to the DSCP value inside the data packets. In this case, DSCP of the data packets may be previously set to give different priorities. These DSCP values of data packets should be defined less than CS4. In the other case, DSCP of the data packets may not be set; thus, all data packets are treated equally. These DSCP values of OpenFlow messages and data packets will be then considered by a packet scheduler to schedule the traffic according to their priorities (such as using WFQ).

4.3 Queue and Packet Scheduler

Multiple queues and a packet scheduler are last two components of PSO modules. Instead of FIFO drop-tail queuing, PSO modules provide a queuing mechanism that can prioritize different traffic. Weighted Fair Queue (WFQ) or Priority Queues (PQ) or min rate or other suitable queues can be deployed for this purpose. The multiple queues are one or more internal queues, attached to a specific port (an SDN interface). These internal queues are defined by network administrators, to schedule out packets from the SDN interface.

After passing through the *traffic classifier*, packets will be differentiated according to the rules in PMT. After that, the DSCP values of the packets may be set (marked), or a queue ID may be specified. For the first case, the *packet scheduler* will schedule the packets according to the DSCP values and scheduling mechanisms (defined by the network administrator). For the second case, the *packet scheduler* will map the specified queue ID directly to a specific internal queue.

For example, the *traffic classifier* may specify traffic priorities by marking DSCP values of the IP header. These DSCP values can provide up to 64 traffic categories without an extra-overhead. The traffic scheduler can then use WFQ to handle different traffic priorities. In the other way, the traffic classifier may specify queue ID, and the packet scheduler then uses PQ or min rate for different traffic types.

DSCP is originally deployed for Quality of Service (QoS) issues. However, this work mainly focuses on mitigating the delay of higher priority traffic only, not covering all QoS parameters.

4.4 Configuration of Policy Map Table

In general, a controller can set or query configuration parameters in an OpenFlow switch using the OpenFlow configuration messages (OFPT_SET_CONFIG, OFPT_GET_CONFIG). In this paper, our PMT is defined by a network administrator at the controller, and distributed to OpenFlow switches using an OFPT_SET_CONFIG message during the connection setup. The controller and switches then have the same PMT for PSO switch controller modules. The configuration steps are as follows:

1) The administrator configures the controller paths for all switches.
2) The administrator creates rules in PMT according to the organization policy at the controller.
3) The controller sends and updates PMT to all switches by using OFPT_SET_CONFIG messages.
4) The administrator can check the PMT of any switch by using OFPT_GET_CONFIG messages.

5. Performance Evaluation

The performance evaluation of our proposed PSO is conducted using the Network Simulation 3 (ns-3) [24]. Chaves's OpenFlow module version 1.3 for ns-3 [25] is also deployed in the simulation. So, our PSO modules are extended from this module to provide a special queue in control channel (SDN interface). All nodes (source and receiver nodes, cross traffic nodes) implement FIFO scheduling and drop-tail queuing. Each simulation is run 50 times using a different Random Number Generator (RNG) seeds to get the averaged results, quoted with error bars with respect to confidence intervals of 95%.

5.1 Performance Metrics

(1) Delay

Delay is a crucial index of the operation efficiency of an SDN network, especially for real-time applications (such as Voice over IP). There have been several studies on delay in SDN, such as [7], [10], [11], [22], [26]. Delay measurement of our study is based on these previous studies. The details are described as follows.

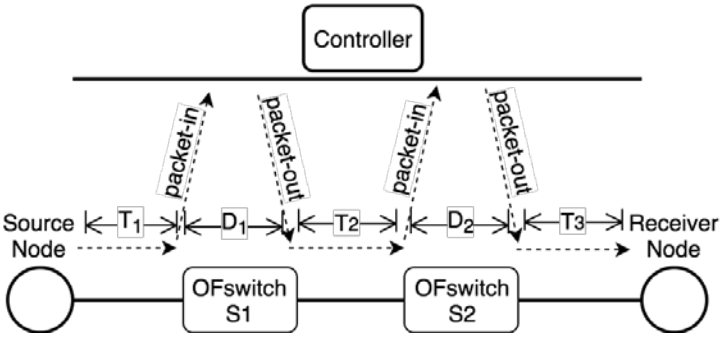

Fig. 5. Delay measurement

As shown in **Fig. 5**, delay of data packet can be measured as the summation of transmission time (T_1+T_2+T_3), and delay in each switch or each hop delay (D_1+D_2). T_1 is a transmission time, counting the time from the source node to the switch S_1. D_1 is the hop delay, counting the time from switch S1 sending a packet-in message to the controller until switch S1 receiving a packet-out message (i.e. acquiring forwarding rules). This includes processing time at both switch S1 and the controller, queuing delay at both switch S1 and the controller, and transmission time of the packet-in and packet-out messages. For the next hop, T_2 is the

transmission time from switch S1 to switch S2. The controller can look at the network end-to-end while making instruction for the switches because it has full topology physical and logical views of the network. So, flow rules of switch S2 are known. These rules can be installed automatically [27]. So, the time to acquire forwarding rules will be excluded from D_2. For this reason, D_2 is obtained by summation of processing delay in the flow-tables (TCAM packet matching delay) and queuing delay in switch S2. Experimental results of this work will be evaluated in term of this delay.

(2) Packet Loss

Packet loss is defined as fraction of the total transmitted packets that did not receive at the receiver. In this work, packet loss is described as the percentage of packets lost with respect to packets sent. Packet loss is generally caused by network congestion. In SDN, packet loss in control links directly affect to data traffic.

(3) Throughput

Throughput is defined as the rate of successful packets delivered over a communication channel. Throughput is usually measured in bits per second (bps). In SDN, network congestion in control links can reduce the throughput of both control and data traffic.

5.2 Network Scenarios

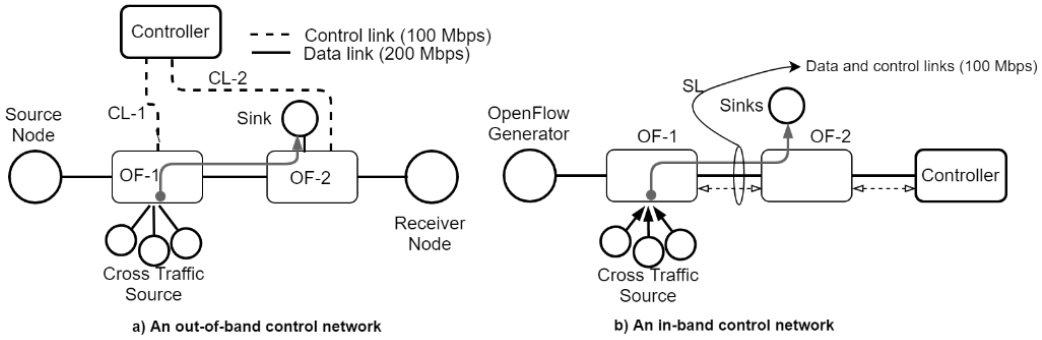

Fig. 6. Network scenarios: a) an out-of-band control network, b) an in-band control network

In the out-of-band control network scenario (as shown in **Fig. 6a**), we define a PMT as shown in the Example-1 (in section 4.2). The objective of this experiment is to test how the increased load on a control link (CL-1) impacts to data traffic, and to test the PSO in terms of throughput and delay of a specific data traffic (high priority traffic). Each link of data traffic has a capacity of 200 Mbps. Each link of control traffic has a capacity of 100 Mbps. A specific data traffic is set to 1000 Kbps, sent from source node to receiver node. To make the competition among OpenFlow messages, switch OF-1 has cross traffic. Cross-traffic nodes generate several data flows and send them via switch OF-1 to the sink node. In this case, switch OF-1 will generate OpenFlow packet-in messages (associated with the data flows), which increase a load on CL-1.

Since the cross-traffic has increased, several OpenFlow packet-in messages are sent to the controller. In this case, the load on a control link (CL-1) (as shown in **Fig. 6a**), is then increased. Therefore, we define this load on CL-1 as Normalized Load (NL), and NL can be obtained as follows:

$$NL = \frac{Data\ rate\ (bps)}{Link\ capacity\ (bps)}$$

In the in-band control network scenario (as shown in **Fig. 6b**), we define a PMT as shown in the Example-2 (in section 4.2). The objective of this experiment is to test how the increased load on a shared-link (SL) impacts to control traffic, and to test if PSO can help mitigate the problems in the in-band control network. The purpose of our experiments in the in-band control network is only to simulate and evaluate the effects of PSO modules. Several mechanisms for the in-band control network (as further described in Section 5.4) are not in the scope of this paper. We have not intended to implement the complete mechanism of the in-band control network. SL has a limited capacity of 100 Mbps. Data and control traffic share the same network link (SL). OpenFlow messages in our experiments are set according to OpenFlow specification 1.3, including 105 bytes of *OpenFlow Extensible Match fields*. Each message is approximately 220 bytes including TCP/IP header. From **Fig. 6b**, OpenFlow generator generates and sends OpenFlow messages via switch OF-1 and switch OF-2 to the controller. Cross-traffic nodes generate several data packets and send them via switch OF-1 to the sink node.

Since the data packets has increased, the load on a SL (as shown in **Fig. 6b**), is then increased. Therefore, we also define this load on SL as Normalized Load (NL) as previously mentioned.

5.3 Simulation Results

(1) Out-of-band control network scenario

Fig. 7a shows packet loss of OpenFlow packet-in and packet-out messages of high priority data traffic, comparing between OpenFlow with PSO and traditional OpenFlow. **Fig. 7b** shows the hop delay in switch OF-1 of high priority data traffic, comparing between OpenFlow with PSO and traditional OpenFlow.

Under a low and medium NL ($NL \leq 0.8$) over CL-1, the results have shown low OpenFlow packet loss (0%) and low delay (2.2 ± 0.6 ms) of both OpenFlow with PSO and traditional OpenFlow. However, at a high load ($NL > 0.8$), the congestion cause a significantly high packet loss (5 ± 1.4 %) in traditional OpenFlow. In this case, as the load increases, a switch drops more OpenFlow packet-in messages. After dropping, a switch has to retransmit these messages after their timeouts. This finally increases hop delay in switch OF-1 (83 ± 2.7 ms). Yet, even with a high load ($NL > 0.8$), OpenFlow with PSO provides a lower OpenFlow packet loss, and a lower hop delay (22 ± 2 ms) of switch OF-1, as shown in **Fig. 7b**.

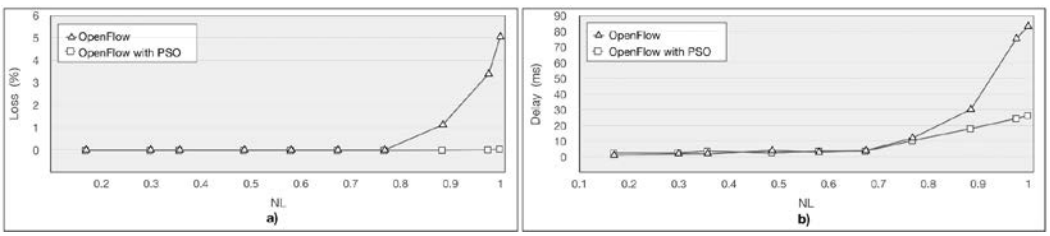

Fig. 7. OpenFlow packet of a high priority data traffic on a control link (CL-1): a) Packet loss, b) Delay

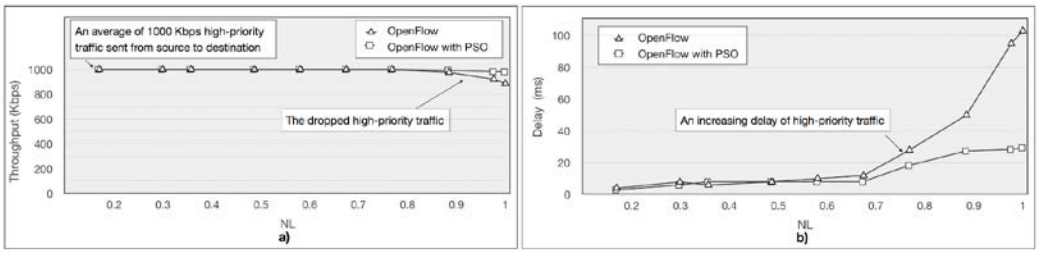

Fig. 8. Impact on a high priority data traffic on a control link (CL-1): a) Throughput, b) Delay

Fig. 8 shows throughput and delay of a high priority data traffic by comparing between OpenFlow with PSO and traditional OpenFlow. In traditional OpenFlow, at a high load (NL > 0.8), some buffered high priority packets are then dropped after their timeouts because the OpenFlow control messages at CL-1 are dropped. So, the throughput is reduced to 887 ± 52 Kbps, and the delay of a high priority data traffic is increased to 103 ± 5 ms. However, in the OpenFlow with PSO even at a high load (NL > 0.8), a higher throughput (992 ± 6 Kbps) of the real-time data traffic can be provided. The PSO can also provide a low delay (27.6 ± 2.8 ms) in comparison to the traditional OpenFlow.

So, our PSO can help the data flow with high priority acquire forwarding rules with lower delay under network congestion at the control link. In case of the congestion at the control link, traditional OpenFlow would cause a severe problem to the delay sensitive services, such as for Voice over IP.

(2) In-band control network scenario

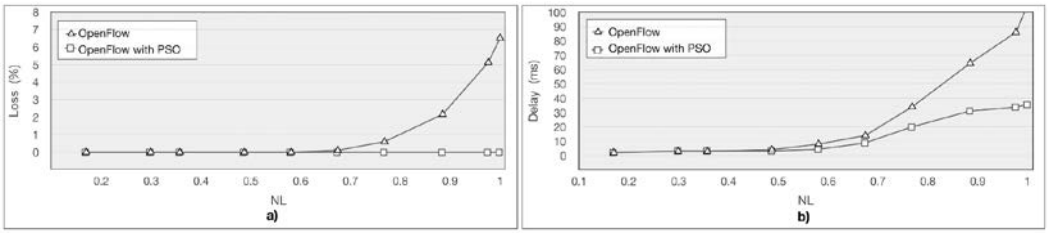

Fig. 9. OpenFlow packet on a shared-link (SL): a) Packet loss, b) Delay

Fig. 9a shows packet loss of OpenFlow messages, comparing between OpenFlow with PSO and traditional OpenFlow. **Fig. 9b** shows delay of the OpenFlow messages, comparing between OpenFlow with PSO and traditional OpenFlow. Under a low NL (NL ⩽ 0.6) over SL, the results have shown low OpenFlow packet loss (0%) and low delay (3 ± 0.4 ms) of both OpenFlow with PSO and traditional OpenFlow. However, at a medium and high load (NL > 0.6), the congestion between OpenFlow and data packets cause a significantly high packet loss (6.4 ± 1.5 %) in traditional OpenFlow. In this case, as the load increases, a switch drops more OpenFlow messages. After dropping, a switch has to retransmit these messages after their timeouts. This finally increases delay in acquiring forwarding rules (> 100 ms). Yet, even with a medium and high (NL > 0.6), OpenFlow with PSO provides a lower OpenFlow packet loss, and a lower delay (32 ± 4 ms), as shown in **Fig. 9b**. Hence, PSO is helpful in the in-band control network to prioritize OpenFlow messages over data packets. This prioritization helps mitigate the packet loss and delay for OpenFlow messages during network congestion.

5.4 Implementation Issues of In-band Control Network

For the in-band control network, we have designed PSO to prioritize OpenFlow messages over data traffic. As previously shown in this paper, the PMT of PSO can be set to prioritize control traffic (OpenFlow messages) over data traffic. This prioritization helps overcome the bandwidth competition between control and data traffic during network congestion. However, OpenFlow mechanisms for in-band control network are still at their early stage. Our in-band simulation is merely focused on PSO implementation without completing the whole design of in-band related mechanisms. There are quite a few open-research issues to complete the design, such as bootstrapping mechanisms (to establish a communication path between switches and a controller), topology discovery mechanisms (to find the most suitable path from a switch to the controller), control path recovery mechanisms (to recover from the control path failure). Some studies (such as [9], [28]–[30]) have initially investigated on the issues but still unsolved. Therefore, we plan for the future work to study further on the aforementioned issues of the in-band control network, then extend our ns-3 modules to perform further experimental evaluation.

5.5 The Analysis of PMT Overhead

According to TCAM operations, the OpenFlow module in ns-3 [25] considers the concept of virtual TCAM to estimate the average flow table search time. To provide a more realistic delay, this module uses sophisticated search algorithms for packet matching such as binary tree. For our design, this algorithm is used for packet matching in the PMT. So, the following equation can be used to estimate the delay of flow tables after adding the delay of the PMT:

$$TCAM\ processing\ time = K \times \log_2(n \times m)$$

where K is the time for a single TCAM operation; n is the number of entries on pipeline flow tables; and m is the number of rules in the PMT.

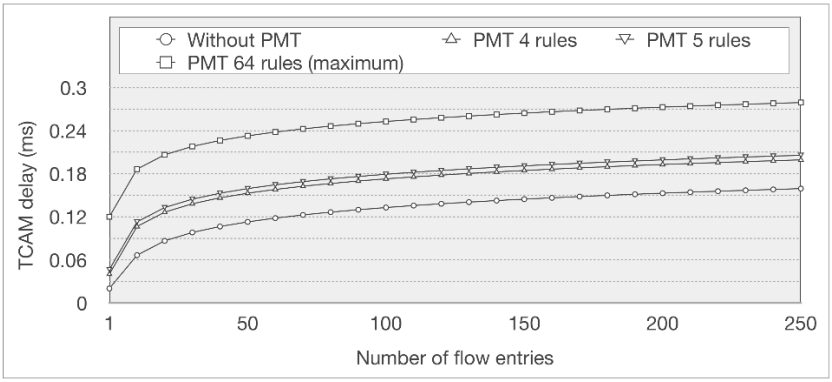

Fig. 10. Flow matching delay

For our PSO, the PMT adds some overhead in a switch. However, this overhead is applicable for all situations (of examples 1-3 in Section 4.2). For example, the maximum rule of these examples is five rules (m=5). According to Chavas [25], K is to set 20 µs. If flow tables have the minimum rule (n=1), the overhead of the flow tables plus PMT could be obtained: $20 \times \log_2(1 \times 5) \approx 46$ µs. **Fig. 10** shows the values of TCAM delay for more

rules in PMT (including 64 rules), and more flow entries in the flow tables. The overall TCAM-delay overhead is acceptable (less than 0.3 ms).

6. Related Work

In this section, we discuss our work in comparison with other studies in the literature. First of all, the standard OpenFlow specification [14] provides only a simple queuing mechanism (i.e., FIFO) to manage the communication between controllers and switches. In in-band control network, an OpenFlow messages may compete with data packets, and finally causes transmission delay of data traffic in acquiring forwarding rules. In out-of-band control network, the OpenFlow control messages of delay-sensitive traffic cannot gain low delay in competing with the OpenFlow control messages of delay-tolerant traffic. A few studies [7]–[9] have worked around this problem. They are discussed as follows.

He et al. [8] have proposed to reduce a delay by redirecting unmatched packets to a proxy. However, using a proxy in SDN may suffer from network congestion if several switches connect to the same proxy. Russ et al. [18] have pointed that a scale and speed are major problems of single point connection (like a proxy). Therefore, adding proxy to solve latency problem may become unmanageable and unavailable. Furthermore, this work is only designed specifically for the out-of-band control network, and cannot be deployed for the in-band control network. Unlike this work, our PSO has been designed to support both out-of-band and in-band control networks. Also, our PSO requires no proxy.

Sharma et al. [9] have proposed to serve traffic with different priorities only in the in-band control network by redirecting packets through Open vSwitch [31] queues that can separated control traffic for data traffic. Yet, this work has not designed to prioritize different types of control messages. The mechanism of this work also needs vendor specific options to handle queue priority. Moreover, this work is only designed for the in-band control network, and cannot be deployed for the out-of-band control network.

Long et al. [7] have proposed to optimize OpenFlow protocol by appending a priority tag to the OpenFlow packet-in message, and adding the Priority-based Flow Rule Request Message Processing Mechanism (PFRRMPM) at the switches and controllers. Similar to ours, this work aims to help the data flow with delay sensitivity acquire the forwarding rule with shorter waiting latency, when there are excess flow rule request messages in the SDN. However, by adding priority tag to the OpenFlow packet-in messages, this work would significantly cause an overheard to the size of the control messages. In contrast, our work does not have such an overhead since it uses the existing DS field in the standard IP header of the control messages for priority marking. In comparison to ours, this work aims to support both in-band and out-of-band control networks as same as our work. Yet, this work has no detail of how to classify different traffic priorities in their design. It is only an initial design, with no prototype. There has been no experiment and performance evaluation to test their design. Our work has proposed more details for prioritizing different traffic, and different types of OpenFlow messages. We have also implemented the prototype of our design. Furthermore, experiments and performance evaluations have been done to demonstrate the success of our design in the out-of-band control network. We also expect positive results in the in-band control network.

7. Conclusions and Future Work

OpenFlow has been deployed to provide benefits of SDN network. However, extra delay is introduced by its flow setup process. From the literature review, this may cause some delay sensitive services to be failed. So, we proposed the PSO modules in both OpenFlow switches and controllers to solve the problem. These mechanisms could overcome the problem of competition among control traffic in the out-of-band control network. Therefore, our approach helped the delay-sensitive traffic to get forwarding in time. The experimental results in out-of-band control network had demonstrated the success of our design. Our PSO design could also mitigate the competition between control and data traffics for in-band control network. The initial results of the in-band environment have demonstrated that OpenFlow with PSO could prioritize OpenFlow messages over data traffic. So, OpenFlow with PSO could gain lower packet loss and delay in comparison with the traditional OpenFlow. For the future work, we plan to add more mechanisms to alleviate the other open-research problems in the in-band control network. The OpenFlow module for ns-3 will be extended, and then experimented further for the in-band environment.

Acknowledgement

This research is funded by the Royal Society Newton Mobility Grant (No: NI160138) from the UK's Official Development Assistance together with Office of Higher Education Commission (OHEC) Thailand, University of Leeds (UK), and Mahasarakham University (Thailand). We are also grateful to a few members of Distributed Systems & Services (DSS) research group (Leeds University, UK) for their comments and discussions.

References

[1] B. Theophilus and A. Aditya, "Unraveling the Complexity of Network Management," in *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation*, Berkeley, CA, USA, pp. 335–348, April 2009. https://dl.acm.org/citation.cfm?id=1559000

[2] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 493–512, First Quarter 2014. Article (CrossRef Link).

[3] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The SoftRouter Architecture," in *Proc. of ACM SIGCOMM Workshop on HOTNETS*, 2004. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.468.5468

[4] A. Doria *et al.*, "Forwarding and Control Element Separation (ForCES) Protocol Specification," IETF RFC-5810, March 2010. https://tools.ietf.org/html/rfc5810

[5] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, March 2008. Article (CrossRef Link).

[6] "ONF: Open Networking Foundation." [Online]. Available: https://www.opennetworking.org.

[7] X. Long, W. Wang, X. Gong, X. Que, and Q. Qi, "Priority based Flow Rule Request Message Processing Mechanism in the OpenFlow Switch," *IETF Internet-Draft*, October 2016. https://tools.ietf.org/html/draft-long-sdnrg-pfrrmpm-openflow-01

[8] K. He, J. Khalid, S. Das, A. Akella, L. Li, and M. Thottan, "Mazu: Taming Latency in Software Defined Networks," *University of Wisconsin-Madison Technical Report*, 2014. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.659.3257

[9] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "In-band control, queuing, and failure recovery functionalities for openflow," *IEEE Network*, vol. 30, no. 1, pp. 106–112, February 2016. Article (CrossRef Link).

[10] S. Sharma *et al.*, "Implementing Quality of Service for the Software Defined Networking Enabled Future Internet," in *Proc. of the Third European Workshop on Software Defined Networks*, London, UK, pp. 49–54, September 2014. Article (CrossRef Link).

[11] K. He *et al.*, "Latency in Software Defined Networks: Measurements and Mitigation Techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 435–436, June 2015. Article (CrossRef Link).

[12] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A Survey of Programmable Networks," *SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, April 1999. Article (CrossRef Link).

[13] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layer and Architecture Terminology," *IETF RFC-7426*, January 2015. https://tools.ietf.org/html/rfc7426

[14] Open Network Foundation, "OpenFlow Switch Specification version 1.5.1 (Protocol version 0x06)," March-2015. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[15] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 116–123, July 2014. Article (CrossRef Link).

[16] S. Jain *et al.*, "B4: Experience with a Globally-deployed Software Defined WAN," in *Proc. of the ACM SIGCOMM*, New York, USA, pp. 3–14, October 2013. Article (CrossRef Link).

[17] L. Schiff, S. Schmid, and P. Kuznetsov, "In-Band Synchronization for Distributed SDN Control Planes," *SIGCOMM Computer Communication Review*, vol. 46, no. 1, pp. 37–43, 2016. Article (CrossRef Link).

[18] W. Russ and Z. Shawn, "Cloudy-Eyed: Complexity and Reality with Software-Defined Networks," *The Internet Protocol Journal*, vol. 19, no. 3, pp. 33–44, November 2016. http://ipj.dreamhosters.com/wp-content/uploads/issues/2016/ipj19-3.pdf

[19] K. Nichols, S. Blake, and F. Baker, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," IETF RFC-2474, December 1998. https://tools.ietf.org/html/rfc2474

[20] P. Almquist, "Type of Service in the Internet Protocol Suite," IETF RFC-1349, July 1992. https://tools.ietf.org/html/rfc1349

[21] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, January 2015. Article (CrossRef Link).

[22] T. Hsiao and W. Chang, "Network controller for delay measurement in SDN and related delay measurement system and delay measurement method," *US9419878-B2*, August 2016. https://patents.google.com/patent/US9667518

[23] J. Benjamin, L. Niels, and A. Kuipers, "Scalability and Resilience of Software-Defined Networking: An Overview," *CoRR*, vol. abs/1408.6760, 2014. https://arxiv.org/abs/1408.6760

[24] "ns-3." [Online]. Available: https://www.nsnam.org.

[25] J. Chaves, C. Garcia, and R. Madeira, "OFSwitch13: Enhancing Ns-3 with OpenFlow 1.3 Support," in *Proc. of the Workshop on Ns-3*, New York, USA, pp. 33–40, June 2016. Article (CrossRef Link).

[26] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang, and H. V. Madhyastha, "Software-Defined Latency Monitoring in Data Center Networks," in *Proc. of Passive and Active Measurement*, vol. 8995, Lecture Notes in Computer Science, pp. 360–372, March 2015. Article (CrossRef Link).

[27] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford, "A NICE Way to Test OpenFlow Applications," in *Proc. of the USENIX Symposium on Networked Systems Design and Implementation*, San Jose, USA, pp. 127–140, April 2012. https://dl.acm.org/citation.cfm?id=2228312

[28] M. Canini, I. Salem, L. Schiff, E. M. Schiller, and S. Schmid, "A Self-Organizing Distributed and In-Band SDN Control Plane," in *Proc. of International Conference on Distributed Computing Systems*, Atlanta, USA, pp. 2656–2657, June 2017. Article (CrossRef Link).

[29] A. S. Tan *et al.*, "Automatic topology discovery in software defined networks," in *Proc. of Signal Processing and Communications Applications Conference*, pp. 939–942, April 2014. Article (CrossRef Link).

[30] S.-C. Lin, P. Wang, and M. Luo, "Control traffic balancing in software defined networks," *Computer Networks*, vol. 106, pp. 260–271, September 2016. Article (CrossRef Link).

[31] "Open vSwitch." [Online]. Available: http://openvswitch.org.

Piyawad Kasabai is a PhD student of Information Technology and Advanced Network (ISAN) research group, Mahasarakham University, Thailand. He received MSc in Information Technology from Mahasarakham University. His research interests include computer network, software-defined network, and network security

Karim Djemame received Ph.D. from University of Glasgow, UK, in 1999, and is currently holding a Professor position at the School of Computing, University of Leeds. He sits on a number of international program committees for cloud middleware. He is currently the scientific and technical manager of the H2020 EU project TANGO. His main research areas focus on Grid/Cloud computing, including system architectures, resource management, and energy efficiency.

Somnuk Puangpronpitag received Ph.D. from University of Leeds, UK. He is currently the director of Information Technology and Advanced Network (ISAN) research group, Mahasarakham University, Thailand. His research interests include network performance engineering, network security and future internet technologies.