Journal of Network and Computer Applications 130 (2019) 76–88

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Performance modeling and analysis of TCP and UDP flows over software defined networks

Yuan-Cheng Lai [a,*], Ahsan Ali [b], Md. Shohrab Hossain [b], Ying-Dar Lin [c]

[a] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[b] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[c] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Keywords:
Software defined networking (SDN)
Performance modeling
Queueing model
TCP
UDP

ABSTRACT

Software Defined Networking (SDN) decouples the control plane from the data plane, thereby facilitating network virtualization, dynamic programmability and flexibility in network management. Previous studies on SDN modeling focus only on packet-level arrivals. However, if flow-level arrivals are not also considered, the model cannot properly reflect the true probability of packets being sent to the controller. Accordingly, the present study proposes two analytical models for predicting the performance of TCP and UDP flows over SDN, respectively, given the assumption of both flow-level arrivals and packet-level arrivals. In constructing the models, the switch and controller are considered jointly and four-dimensional states are used to evaluate the steady-state probabilities of the states. Analytical formulae are derived for the average packet delay and packet loss probability of the TCP and UDP flows. Simulation results very well match with the analytical ones, thereby validating our analytical models. The results show that TCP significantly outperforms UDP over SDN architectures. In particular, TCP reduces the packet delay by 12 ∽ 50% and the packet loss probability by 25 ∽ 100%.

1. Introduction

Software Defined Networking (SDN), a new network paradigm, is seen as a promising approach for resolving the technological obstinacy of networks in the future (Rowshanrad et al., 2014; Hakiri et al., 2014; Pan et al., 2011). SDN has already proven effective in improving the network performance in data centers (Pries et al., 2012; Bari et al., 2013). For example, Google implemented SDN to run its data center WANs as far back as January 2012 (Holzle, 2012). SDN not only enables improved flexibility in network routing, but also provides the means to change the behavior of one part of a network by isolating it from the other parts (Jarschel et al., 2011). Moreover, SDN facilitates dynamic programmability for forwarding packets in highly distributed networks (ONF, 2012; ONF, 2013). It also facilitates the implementation of network virtualization (Drutskoy et al., 2013; Jain and Paul, 2013), user mobility, energy saving (Heller et al., 2010), and new network and transport layer protocols (Jarschel et al., 2011). All of these advantages are made possible by the main architectural principle of SDN, namely the disassociation of the control plane from the data plane (ONF, 2012; ONF, 2013).

SDN has three cooperative planes: the data plane, the control plane and the application plane. The data plane contains mostly switches, which provide a packet forwarding service under the instructions of the control plane (Xiong et al., 2016). The control plane maintains the overall network information and provides optimized routing for the packets in the data plane. Finally, the control plane communicates with the data plane through a southbound interface known as OpenFlow (McKeown et al., 2008). The control plane also uses northbound interfaces, such as Restful APIs, to interact with the application plane. In the SDN architecture, the controller (located in the control plane) is logically centralized and thus has a global view of the entire network. Importantly, this not only enables the controller to properly manage the bulk network traffic, but also provides software engineers with the ability to customize and scale the network in accordance with changes in the network traffic load and the particular features and services required in the application layer.

SDN has two operating modes, namely a proactive mode and a reactive mode. In the proactive mode, the controller pre-installs a set of flow rules on each switch such that it can forward incoming packets (from the sender) toward the destination as soon as they are received

* Corresponding author.
E-mail addresses: laiyc@cs.ntust.edu.tw (Y.-C. Lai), ahsanali.buet@gmail.com (A. Ali), mshohrabhossain@cse.buet.ac.bd (Md.S. Hossain), ydlin@cs.nctu.edu.tw (Y.-D. Lin).

https://doi.org/10.1016/j.jnca.2019.01.010
Received 29 June 2018; Received in revised form 30 December 2018; Accepted 12 January 2019
Available online 21 January 2019

without any further involvement of the controller. The proactive mode is used mainly in large-scale networks, such as data centers where most of the traffic in the network topology is known in advance. In the reactive mode, the controller installs the flow rule for each packet only after it is forwarded to the controller from the switch. The reactive mode is thus ideal for highly dynamic networks, in which the network topology undergoes frequent change. Packets in the reactive mode experience longer delays than those in the proactive mode. Consequently, the problem of developing mathematical models for the SDN performance in the reactive mode has attracted particular attention in the literature (Jarschel et al., 2011; Mahmood et al., 2014, 2015; Goto et al., 2016; Metter et al., 2016; Fahmin et al., 2018).

Many studies have performed experimental or numerical investigations into the performance of SDN architectures. However, in practice, analytical modeling tends to provide a more time efficient approach; particularly for large-scale networks. Several initial attempts have been made to develop analytical models for the SDN performance based on feedback-oriented queueing theory (Jarschel et al., 2011; Mahmood et al., 2014, 2015). In these models, the switch queue is modeled as M/M/1, while the controller queue is modeled as M/M/1/m. However, all of the models (Jarschel et al., 2011; Mahmood et al., 2014, 2015) assume the switch buffer size to be infinite and consider only packet level arrivals. Also the propagation delay between the switch and the controller is ignored; thereby introducing significant analytical errors.

The authors in Bozakov and Rizk (2013) and Azodolmolky et al. (2013a, 2013b) developed mathematical models based on network calculus to obtain the upper bound of the transmission latency and buffer size at the SDN controller and switch. However, while these models provide the upper-bound performance in the worst case, they provide no clue as to the average performance in the steady state.

Several recent studies (Miao et al., 2015, 2016; Goto et al., 2016) have modeled the SDN data plane using a priority queue-based approach, in which the switches are treated as either M/M/1 (Miao et al., 2015) or MMPP/M/1 (Miao et al., 2016) queueing systems and the controller buffer size is assumed to be either finite (Miao et al., 2015, 2016) or infinite (Goto et al., 2016).

In general, none of the studies above consider *flow*-level arrivals when evaluating the *average* performance of the SDN. Consequently, significant analytical errors inevitably occur. Accordingly, the present study develops two analytical models for predicting the average performance (in the steady state) of two different types of flow[1] (TCP and UDP) over SDN architectures, which consider not only packet-level arrivals, but also flow-level arrivals, i.e., multiple packets per TCP/UDP flow. There are many differences between TCP and UDP. However, one of the main differences is that TCP is connection-oriented and performs a three-way handshake each time a connection is required, whereas UDP is connectionless. This impacts the number of packets sent to the controller in the two cases, and therefore potentially results in significant differences in the packet delay and packet loss probability.

In developing analytical models for the performance of TCP and UDP flows over SDN, the present study considers the reactive SDN mode since, as described above, reactive SDN mode is ideal for the network topology which undergoes frequent change. Furthermore, for reasons of simplicity, the analysis considers only the connection establishment process in the TCP model, and ignores the effects of retransmission, congestion control and flow control. Both models consider the switch and controller jointly and utilize four-dimensional states to evaluate steady-state probabilities of the states. The main *contributions* of this study are threefold. First, analytical models are developed to predict the performance of TCP and UDP flows over SDN. Second, simulations are performed to demonstrate the validity of the proposed models. Third, an analysis is performed to investigate the effects of the main system parameters (i.e., flow arrival rate, flow termination rate, controller service rate) on the SDN performance for both types of flow.

The remainder of this paper is organized as follows. Section 2 introduces previous studies on SDN modeling and describes the TCP and UDP flow scenarios considered in the present study. Section 3 explains the system model and derives the proposed analytical models for TCP and UDP flows over SDN. Section 4 compares the analytical results obtained from the TCP and UDP models with the corresponding simulation results under various parameter settings. Finally, Section 5 provides some brief concluding remarks and indicates the intended direction of future research.

2. Background and related works

This section commences by describing previous related work on SDN modeling. The TCP and UDP flow scenarios considered in the present study are then briefly introduced and explained.

2.1. Related works

Table 1 shows the main studies published in the literature on the analytical modeling of SDN architectures.

Jarschel et al. (2011) used feedback-orientated queueing theory to capture the interaction between the control plane (single controller) and the data plane (single switch), and modeled the switch and controller as M/M/1 and M/M/1/m Markovian queueing systems, respectively. The validity of the proposed model was demonstrated by evaluating the impact of the controller service time on the average packet delay. However, the accuracy of the model was degraded as the probability of new flows increased. Mahmood et al. (2014) overcame this problem by using Jackson assumption to estimate the packet rate from the controller to the switch such that the overall packet arrival rate at the switch could be more reliably obtained. In a later study, the same group (Mahmood et al., 2015) used a Jackson network to model the data plane for the case where the controller was responsible for multiple switches and was modeled as an M/M/1 queue with either a finite or infinite buffer size. The model was used to evaluate the effects of the packet forwarding probability to controller and the controller service time on the average packet delay and network throughput. The authors additionally derived a closed-form expression for the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) of the packet delay for a given path.

Bozakov et al. (Bozakov and Rizk, 2013), used a queueing model to characterize the behavior of the control interface between the controller and the switch in terms of the number of serviced messages over different time scales. A calculus-based approach was used to derive an estimate of the corresponding service curves. A simple interface extension for controller frameworks was additionally proposed to help operators configure the delay bounds for the transmitted control messages. Azodolmolky et al. (2013a) presented a mathematical framework based on network calculus to support the scalable SDN deployment, in which the upper bound of the packet delay and buffer size of the controller were evaluated under the assumption of a cumulative arrival process at the SDN controller. The same group (Azodolmolky et al., 2013b) later extended this work to model the switch performance in terms of the delay and queue length boundaries. However, the frameworks proposed in Bozakov and Rizk (2013) and Azodolmolky et al. (2013a, 2013b) are all based on deterministic network calculus, and hence cannot properly reflect the true performance of SDN when the network is in an equilibrium state.

Wang et al. (2015) proposed a multistage controller for improving the flexibility of the SDN control plane incorporating an M/M/1 root controller and multiple M/M/1/m local controllers. A model was derived to analyze the average packet delay at each controller. How-

[1] Although the term TCP connection is commonly used in the literature, the present study uses the term TCP flow to be consistent with the term UDP flow.

Table 1
Existing works on SDN modeling.

Paper	Component	#	Methodology	Performance Metrics
Jarschel et al. (2011)	Switch	1	M/M/1	Avg. packet delay
	Controller	1	M/M/1/m	
Mahmood et al. (2014)	Switch	1	M/M/1	Avg. packet delay
	Controller	1	M/M/1/m	Network throughput
Mahmood et al. (2015)	Switch	N	M/M/1	Avg. packet delay
	Controller	1	M/M/1, M/M/1/m	Network throughput
Bozakov and Rizk (2013)	Switch	1	Net. Calculus	Packet delay bound
	Controller	1	Net. Calculus	
Azodolmolky et al. (2013a)	Switch	N	Net. Calculus	Buffer size bound
	Controller	1	Net. Calculus	Packet delay bound
Azodolmolky et al. (2013b)	Switch	N	Net. Calculus	Buffer size bound
	Controller	1	Net. Calculus	Packet delay bound
Wang et al. (2015)	Local Controller	N	M/M/1/m	Avg. packet delay
	Root Controller	1	M/M/1	
Miao et al. (2015)	Switch	1	HPQ: M/M/1 LPQ: M/M/1	Avg. packet delay
	Controller	1	M/M/1/m	Packet loss probability
Miao et al. (2016)	Switch	1	HPQ: MMPP/M/1 LPQ: MMPP/M/1/m	Avg. packet delay
	Controller	1	MMPP/M/1/m	Network throughput
Xiong et al. (2016)	Switch	N	M^X/M/1	Avg. packet delay
	Controller	1	M/G/1	Network throughput
Goto et al. (2016)	Switch	1	3-D state	Avg. packet delay
	Controller	1	(controller,HPQ,LPQ)	Packet loss probability
Fahmin et al. (2018)	Switch	1	M/M/1	Avg. packet delay with NFV
	Controller	1	M/M/1	
Our	Switch	1	4-D state	Avg. packet delay
model	Controller	1		Packet loss probability

ever, the model ignored the switches and produced a large delay for packets due to multiple controllers at which the packets were required to wait. Miao et al. (2015) proposed a preemption-based packet-scheduling scheme to improve the global fairness of SDN and reduce the packet loss probability in the data plane. An analytical model was derived to quantitatively evaluate the scheduling scheme and to pinpoint the performance bottleneck in the SDN architecture. In constructing the model, the data plane was assumed to consist of one M/M/1 queue with low priority and one M/M/1 queue with high priority. By contrast, the control plane was modeled using a single M/M/1/m queue. The same group (Miao et al., 2016) later used a Markov Modulated Poisson Process (MMPP) to better model the bursty nature of packet arrivals of multimedia traffic in SDN architectures. As in their previous study (Miao et al., 2015), the data plane was modeled using two priority queues, namely a high-priority MMPP/M/1 queue and a low-priority MMPP/M/1/m queue, and the control plane was modeled using a single MMPP/M/1 queue. The validity of the proposed model was demonstrated by evaluating the average packet delay and network throughput in a typical SDN network.

Xiong et al. (2016) presented a queuing model for OpenFlow networks based on the assumption of a batch (rather than Poisson) arrival of the packets at switches. The packet forwarding of the switches and the packet processing of the controller were thus modeled as M^X/M/1 and M/G/1 queues, respectively. Closed-form expressions were derived for both the average packet delay and the corresponding PDF. Finally, numerical simulations were performed to evaluate the controller performance under various network scenarios.

Goto et al. (2016) proposed a queueing model which took into account the class-full treatment of different packets arriving at the switch. In particular, the switches were assumed to have two finite-buffer queues, namely a high-priority queue for those packets sent back from the controller and a low-priority queue for newly-arrived packets from other switches. Meanwhile, the controller buffer was assumed to have an infinite capacity. The system was thus represented as a three-dimensional state comprising the controller queue length, the high-priority queue length, and the low-priority queue length, respectively. The resulting model was used to derive the packet loss probability at the switch and the average packet delay for a network consisting of packets with three different classes.

In a previous study (Fahmin et al., 2018), the present group proposed two analytical models of SDN integrated with Network Function Virtualization (NFV); one with NFV under the controller and another with NFV aside the controller. The main differences between the present study and that in Fahmin et al. (2018) are as follows: (1) the present models utilize a four-dimensional state to obtain the steady state probability of system states, whereas the previous models (Fahmin et al., 2018) utilized M/M/1 queuing model; (2) the present models consider both flow-level and packet-level arrivals, whereas the previous models (Fahmin et al., 2018) considered only packet-level arrivals; and (3) the present study investigates the performance of TCP and UDP flows over SDN, while the previous study (Fahmin et al., 2018) focused on the integration issues between SDN and NFV.

2.2. TCP and UDP flows in SDN

Fig. 1 illustrates the TCP flow scenario considered in the present study. As shown, the source first sends a connection request (TCP SYN packet) to the destination through the SDN switch. When the TCP SYN packet arrives at the switch, the switch matches the header portion of the packet against its flow table entries. If the flow table contains an entry for the TCP SYN packet, the switch simply forwards the SYN packet to the destination. However, if a corresponding entry cannot be found, the switch forwards the SYN packet to the controller instead. The controller determines an appropriate routing for the packet, updates the switch flow table accordingly and sends the SYN packet back to the switch. On receipt of the packet, the switch consults the updated flow table and then forwards the packet to the destination accordingly. When the destination receives the SYN packet, it returns a request plus acknowledgement (SYN + ACK) packet to the source through the SDN switch. If the switch has a flow table entry for the SYN + ACK packet, it forwards the packet directly to the source; otherwise it forwards it to the controller to update its flow table. Finally, when the source receives the SYN + ACK packet, it sends an ACK packet to the destination through the SDN switch. Through this three-way handshake process, an end to

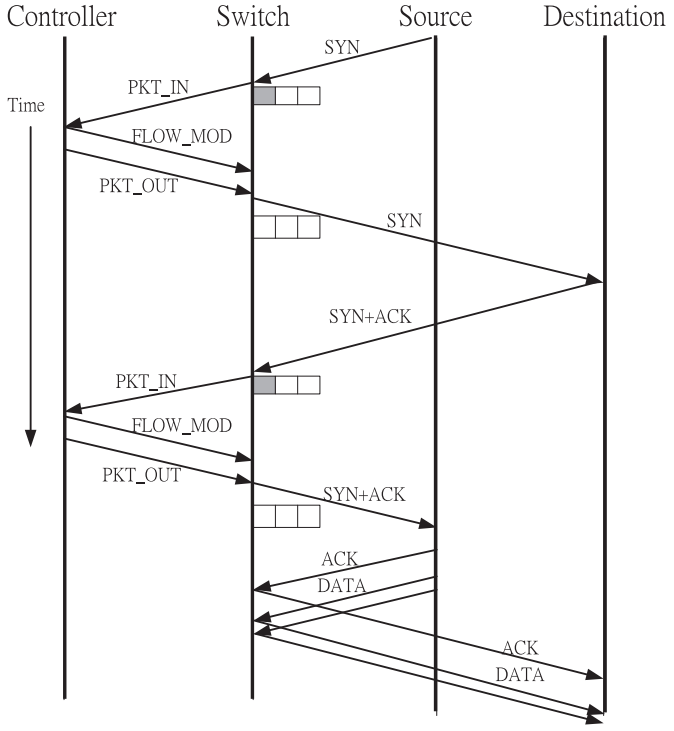

Fig. 1. TCP flow scenario in SDN architecture.

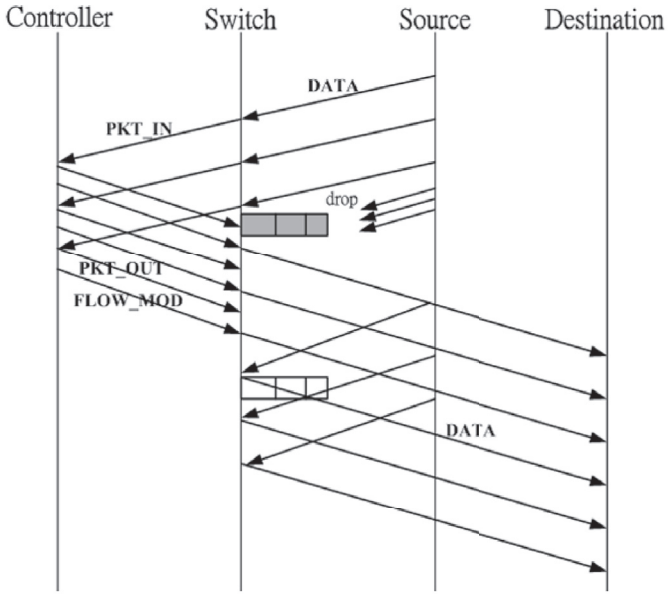

Fig. 2. UDP flow scenario in SDN architecture.

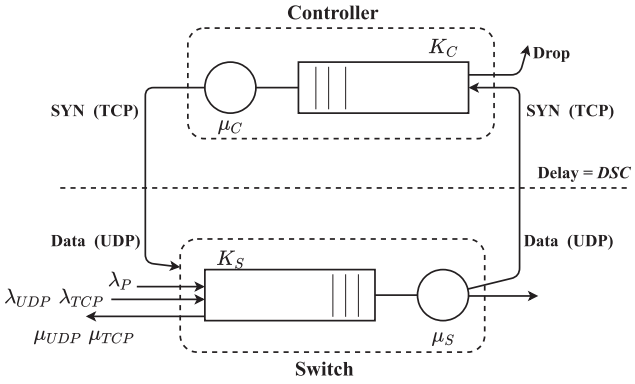

Fig. 3. Queuing model of SDN for TCP and UDP flow.

end connection of TCP flow is established which allows the source to send its data packets to the destination through SDN switch directly without any further involvement of the controller.

Fig. 2 shows the considered UDP flow scenario. When the first data packet of the UDP flow arrives at the switch, it is forwarded directly to the destination if the switch has a corresponding flow table entry, or the data packet is forwarded to the controller otherwise. As the controller processes the data packet, any other packets belonging to the same UDP flow arriving at the switch are also routed to the controller. However, once the switch receives the updated flow table entry from the controller, it sends any subsequent UDP data packets to the destination directly without any further processing by the controller. In both scenarios described above (i.e., TCP and UDP), the switch simply drops any newly-arriving packets in the event that the switch buffer is full.

3. Modeling and analysis

In the present analysis, the switch and controller are considered jointly and are modeled using a continuous-time Markov chain. This section commences by developing a generic SDN queueing model. The detailed SDN TCP and SDN UDP models are then derived and explained.

3.1. Generic queueing model for SDN system

Fig. 3 shows the generic SDN queueing model for TCP and UDP flows consisting of one switch and one controller, where each one has its own queue. To simplify the analysis, the present study imposes the following assumptions.

- The TCP/UDP flow arrivals follow a Poisson process and the packet arrivals within each flow is also a Poisson process. In other words, the flow inter-arrival time and packet inter-arrival time in each flow follow an exponential distribution. As discussed in Wu et al. (2007), Arshadi and Jahangir (2011) and Arfeen et al. (2013), the IP flow inter-arrival time actually follows a Weibull distribution. However, the authors in Jie et al. (2015) confirmed that the flow arrival process in real-world packet switching networks can nevertheless be approximated as a Poisson distribution. The authors in Barakat et al. (2003) also reported that the flow arrivals on Internet backbone links can be well approximated as a Poisson process. Finally, the authors in Metter et al. (2017) also assumed a Poisson process for SDN signaling traffic (i.e., new flow arrivals). Therefore, it seems reasonable to adopt a similar assumption in the present study for the TCP/UDP flow arrivals. In practical networks, the packet arrivals may exhibit a fractal behavior. However, most of the SDN models presented in Table 1 use a Poisson process to model the packet arrivals at the switch. Consequently, the present study also assumes that packets in each flow arrive in accordance with Poisson process, generating the packet arrival rate at the switch as an MMPP process, which is more complex than the Poisson process.
- The TCP/UDP flow duration follows an exponential distribution. This assumption is consistent with that in Metter et al. (2017), Zhang (2005) and Yang et al. (2008), and is adopted in the present study for reasons of simplicity even though it does not strictly hold in practice.
- The packet service time follows an exponential distribution at both the switch and the controller. This assumption is consistent with that

Table 2
Notations used in the queueing model.

Symbol	Description
λ_{TCP}	TCP flow (SYN packet) arrival rate at switch
λ_{UDP}	UDP flow arrival rate at switch
λ_P	Per flow arrival rate at switch
μ_S	Service rate at switch
μ_C	Service rate at controller
μ_{TCP}	TCP flow termination rate
μ_{UDP}	UDP flow termination rate
K_S	Switch queue size
K_C	Controller queue size
DSC	Propagation delay between switch and controller
N_{TCP1}	Number of TCP flows sending SYN packet to the controller
N_{TCP2}	Number of other TCP flows
N_{UDP1}	Number of UDP flows sending data packet to the controller
N_{UDP2}	Number of other UDP flows
q_S	Switch queue length
q_C	Controller queue length

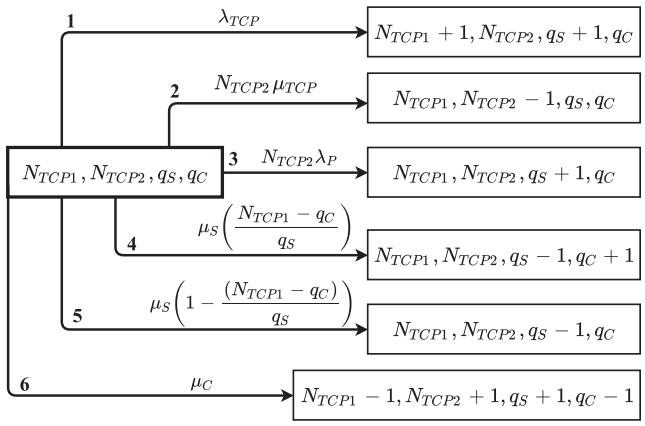

Fig. 4. State transition diagram of the SDN TCP model.

adopted in many previous studies (Jarschel et al., 2011; Mahmood et al., 2014, 2015; Miao et al., 2015, 2016; Goto et al., 2016; Fahmin et al., 2018), and is reasonable since the packet size also follows an exponential distribution.

- The switch buffer size and controller buffer size are both finite. Most previous papers (Jarschel et al., 2011; Mahmood et al., 2014, 2015; Wang et al., 2015; Miao et al., 2015; Fahmin et al., 2018) assume the buffer size to be infinite such that the system can be modeled as an M/M/1 queueing system. However, in practice, the buffers are actually finite and also very few papers (Goto et al., 2016; Miao et al., 2016) considered a finite buffer size for both the switch and the controller in order to properly calculate the packet loss probability.
- In the case of TCP flows, no data packets arrive at the switch before the SYN packet has been handled (i.e., a table entry for the TCP flow has been set up in the switch flow table). This assumption is consistent with the three-way handshake procedure used to establish practical TCP connections, and is hence reasonable.
- In the case of UDP flows, additional data packets can arrive at the switch before the first data packet of the UDP flow has been handled by the controller. Again, this assumption is reasonable since UDP is a connectionless protocol in which the aim is simply to send the data to the destination as quickly as possible.
- For both TCP and UDP, all of the flows are regarded as new flows at the switch.

Table 2 lists the notations used in the queueing model (Fig. 3).

To more accurately determine the packet arrival rate, the TCP flows are classified into two types, namely TCP1 and TCP2, where TCP1 are TCP flows whose SYN packets are being handled by the controller, while TCP2 are all the other TCP flows. As described above, UDP data packets with no flow table entry are redirected by the switch to the controller. For any new UDP flow arriving at the switch, the first data packet will inevitably have a table miss in the switch, and will thus be directed to the controller. For convenience, such data packets are denoted as FIR packets. Accordingly, the UDP flows are also classified into two types, namely UDP1 and UDP2, where UDP1 are UDP flows whose FIR packets are being handled by the controller, and UDP2 are all the other UDP flows. In developing the proposed SDN queueing models, the four-dimensional state $(N_{TCP1}, N_{TCP2}, q_S, q_C)$ is used for the TCP model, while $(N_{UDP1}, N_{UDP2}, q_S, q_C)$ is used for the UDP model.

3.2. SDN TCP model

This subsection begins by describing the transitions of the system states in the TCP model. The related performance metrics are then derived.

1) *State transitions of SDN TCP model*: Fig. 4 shows the six possible transitions (labeled from 1 through 6) of state $(N_{TCP1}, N_{TCP2}, q_S, q_C)$ in the SDN system model for TCP flows. The various transitions are described in the following.
   - First transition: the SYN packet of a new TCP flow arrives at the switch and has no flow table entry (i.e., the packet will be redirected to the controller later). States N_{TCP1} and q_S in the four-dimensional-state model are thus increased by 1. The transition rate is λ_{TCP}.
   - Second transition: a TCP flow terminates. The number of active TCP flows, N_{TCP2}, is decreased by 1. The transition rate is computed as the product of N_{TCP2} and μ_{TCP}.
   - Third transition: a data packet of a TCP flow arrives at the switch. q_S is thus increased by 1. The transition rate is computed as the product of λ_P and N_{TCP2}, where λ_P is the per flow data packet arrival rate at the switch.
   - Fourth transition: the switch redirects a SYN packet (no flow table entry) to the controller. q_S is decreased by 1 and q_C is increased by 1. The probability of redirecting a SYN packet to the controller is equal to $(N_{TCP1} - q_C)/q_S$. Hence, the transition rate is $((N_{TCP1} - q_C)/q_S)\mu_S$.
   - Fifth transition: a SYN packet (or data packet) with a flow table entry arrives at the switch and is forwarded to the destination. q_S is thus decreased by 1. The probability of forwarding a SYN packet (with a flow table entry) or data packet to the destination is $(1 - (N_{TCP1} - q_C)/q_S)$. Consequently, the transition rate is computed as $(1 - (N_{TCP1} - q_C)/q_S)\mu_S$.
   - Sixth transition: the controller serves a SYN packet and forwards it to the switch. q_C is decreased by 1 and q_S is increased by 1. In addition, N_{TCP1} is decreased by 1 and N_{TCP2} is increased by 1. The transition rate is μ_C.
2) *Performance metrics of SDN TCP model*: The mean queue length at the switch can be determined as

$$\overline{q_S^T} = \sum x \times P(N_{TCP1}, N_{TCP2}, q_S = x, q_C). \quad (1)$$

Meanwhile, the mean queue length at the controller can be determined as

$$\overline{q_C^T} = \sum y \times P(N_{TCP1}, N_{TCP2}, q_S, q_C = y). \quad (2)$$

Note that $P(N_{TCP1}, N_{TCP2}, q_S, q_C)$ is the steady state probability of state $(N_{TCP1}, N_{TCP2}, q_S, q_C)$ in the SDN TCP model. The packet delay at the switch, D_S^T, can be calculated as

$$D_S^T = \left(\frac{\overline{q_S^T} + 1}{\mu_S}\right). \quad (3)$$

Similarly, the packet delay at the controller, D_C^T, can be calculated as

$$D_C^T = \left(\frac{\overline{q_C^T} + 1}{\mu_C} \right). \tag{4}$$

In the SDN system model considered in the present study, the incoming packets at the switch are simply dropped if the switch queue is full (i.e., $q_S = K_S$). Hence, the packet loss probability at the switch, L_S^T, can be determined as

$$L_S^T = \sum P(N_{TCP1}, N_{TCP2}, q_S = K_S, q_C). \tag{5}$$

The incoming packets at the controller are also dropped when the controller queue is full (i.e., $q_C = K_C$). Thus, the packet loss probability at the controller, L_C^T, is obtained as

$$L_C^T = \sum P(N_{TCP1}, N_{TCP2}, q_S, q_C = K_C). \tag{6}$$

(I) Average delay of SYN packets, D_{SYN}^T:
In the SDN TCP model, the SYN packets are queued twice, i.e., once at the switch and once at the controller. The SYN packet delay at the switch is similar to D_S^T, while the SYN packet delay at the controller is similar to D_C^T. Hence, D_{SYN}^T can be computed simply as $D_C^T + 2 \times D_S^T + 2 \times DSC$. In other words, D_{SYN}^T is determined as

$$D_{SYN}^T = \left(\frac{\overline{q_C^T} + 1}{\mu_C} \right) + 2 \left(\frac{\overline{q_S^T} + 1}{\mu_S} \right) + 2DSC. \tag{7}$$

(II) Average delay of data packets, D_{data}^T:
In the SDN TCP model, the data packets are not redirected to the controller. Consequently, the average data packet delay, D_{data}^T, is similar to the packet delay at the switch, D_S^T. In other words,

$$D_{data}^T = D_S^T. \tag{8}$$

(III) Loss probability of SYN packets, L_{SYN}^T:
In the SDN TCP model, the SYN packets may be dropped at either the switch or the controller. Hence, L_{SYN}^T can be determined as

$$L_{SYN}^T = 1 - (1 - L_S^T)^2 (1 - L_C^T). \tag{9}$$

(IV) Loss probability of data packets, L_{data}^T :
In the SDN TCP model, the data packets are not redirected to the controller. As a result, the data packet loss probability, L_{data}^T, is the same as the packet loss probability at the switch, L_S^T, i.e.,

$$L_{data}^T = L_S^T. \tag{10}$$

3.3. SDN UDP model

This subsection commences by describing the transitions of the system states in the UDP model. The related performance metrics are then derived.

1) *State transitions of SDN UDP model*: Fig. 5 shows the six possible transitions (labeled from 1 through 6) of state $(N_{UDP1}, N_{UDP2}, q_S, q_C)$ in the SDN system model for UDP flows.

The first, second and sixth state transitions are similar to those for the SDN TCP model, and are hence omitted here. The third, forth and fifth state transitions are described in the following.

- Third transition: a data packet of a UDP flow arrives at the switch. q_S is thus increased by 1. The number of active UDP flows is equal to $(N_{UDP1} + N_{UDP2})$. Hence, the transition rate is computed as the product of λ_P and $(N_{UDP1} + N_{UDP2})$.

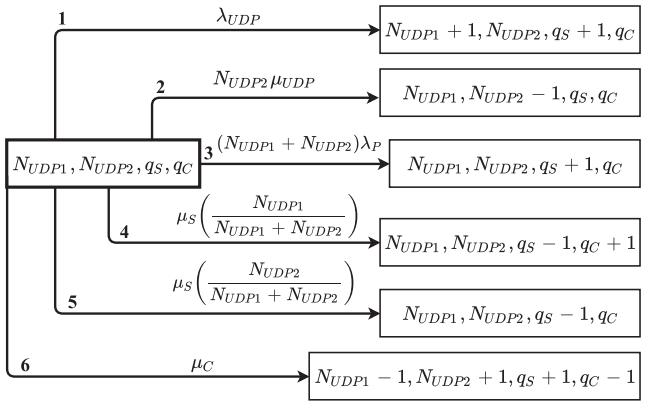

Fig. 5. State transition diagram of the SDN UDP model.

- Fourth transition: the switch serves a data packet with no flow table entry. The data packet is redirected to the controller. Hence, q_S is decreased by 1 and q_C is increased by 1. The probability of a data packet being redirected to the controller is equal to $N_{UDP1}/(N_{UDP1} + N_{UDP2})$. Consequently, the transition rate is computed as $(N_{UDP1}/(N_{UDP1} + N_{UDP2}))\mu_S$.
- Fifth transition: the switch serves a data packet with a flow table entry. The data packet is forwarded directly to the destination. Thus, q_S is decreased by 1. The probability of a data packet being forwarded directly to the destination is equal to $N_{UDP2}/(N_{UDP1} + N_{UDP2})$. Hence, the transition rate is computed as $(N_{UDP2}/(N_{UDP1} + N_{UDP2}))\mu_S$.

2) *Performance metrics of SDN UDP model*: The mean queue length at the switch, $\overline{q_S^U}$, the mean queue length at the controller, $\overline{q_C^U}$, the packet delay at the switch, D_S^U, the packet delay at the controller, D_C^U, the packet loss probability at the switch, L_S^U, and the packet loss probability at the controller, L_C^U, are given respectively as

$$\begin{aligned}
\overline{q_S^U} &= \sum x \times P(N_{UDP1}, N_{UDP2}, q_S = x, q_C), \\
\overline{q_C^U} &= \sum y \times P(N_{UDP1}, N_{UDP2}, q_S, q_C = y), \\
D_S^U &= \left(\frac{\overline{q_S^U} + 1}{\mu_S} \right), \\
D_C^U &= \left(\frac{\overline{q_C^U} + 1}{\mu_C} \right), \\
L_S^U &= \sum P(N_{UDP1}, N_{UDP2}, q_S = K_S, q_C), \\
L_C^U &= \sum P(N_{UDP1}, N_{UDP2}, q_S, q_C = K_C).
\end{aligned} \tag{11}$$

The Average delay of the FIR packets, D_{FIR}^U, is similar to the SYN packet delay, D_{SYN}^T. Moreover, the loss probability of the FIR packets, L_{FIR}^U is similar to the SYN packet loss probability, L_{SYN}^T. In other words,

$$\begin{aligned}
D_{FIR}^U &= \left(\frac{\overline{q_C^U} + 1}{\mu_C} \right) + 2 \left(\frac{\overline{q_S^U} + 1}{\mu_S} \right) + 2DSC. \\
L_{FIR}^U &= 1 - (1 - L_S^U)^2 (1 - L_C^U).
\end{aligned} \tag{12}$$

(I) Average delay of data packets, D_{data}^U:
In the SDN UDP model, some of the data packets are redirected to the controller. These data packets experience a delay at both the switch and the controller. Hence, their delay is equal to D_{FIR}^U. For the other data packets, the delay is similar to D_S^U. Assuming

that the probability of the data packets being directed to the controller is denoted as P_X, the data packet delay in the SDN UDP model, D^U_{data}, can be calculated as

$$D^U_{data} = (1 - P_X) \times D^U_S + P_X \times D^U_{FIR}. \quad (13)$$

In practice, P_X represents the ratio of the number of data packets redirected to the controller to the number of data packets served first time at the switch. In other words, P_X is given by

$$P_X = \frac{\overline{N_{UDP1}}}{(\overline{N_{UDP1}} + \overline{N_{UDP2}})},$$

where $\overline{N_{UDP1}} = \sum x \times P(N_{UDP1} = x, N_{UDP2}, q_S, q_C)$ and $\overline{N_{UDP2}} = \sum y \times P(N_{UDP1}, N_{UDP2} = y, q_S, q_C)$.

(II) Loss probability of data packets, L^U_{data} :
As described above, some of the data packets are redirected to the controller in the SDN UDP model. These packets may be dropped at either the switch or the controller. Consequently, their loss probability is equal to L^U_{FIR}. The loss probability of other data packets is equal to L^U_S. As a result, the data packet loss probability in the SDN UDP model, L^U_{data}, is given as

$$L^U_{data} = (1 - P_X) \times L^U_S + P_X \times L^U_{FIR}. \quad (14)$$

4. Analytical and simulation results

This section presents and discusses the analytical results obtained from the TCP and UDP models for the average packet delay and packet loss probability given various values of the system parameters. For validation purposes, the analytical results are compared with those obtained from simulations.

4.1. Simulation setup

A JAVA custom simulator program capable of supporting both flow-level arrivals and packet-level arrivals was developed based on the queueing model assumptions described in Subsection 3.1. The simulations are briefly described in the following. (Note that full details of the simulation study are available in Lai et al. (2018a) and Lai et al. (2018b), while the simulation code is available in Ali (2018).

The SDN TCP and UDP simulations (Lai et al., 2018a, 2018b) considered seven discrete events: (i) flow packet arrival at the switch from the source, (ii) data packet arrival at the switch from the source, (iii) packet arrival at the switch from the controller, (iv) packet arrival at the controller from the switch, (v) switch service completion, (vi) controller service completion, and (vii) flow termination. The seven events were stored in an event priority queue (*EPQ*) based on their event time (where the event times were updated in accordance with the queueing model described in Section 3.1). Two separate FIFO queues were used to store the SYN/FIR packets (*SFSList*) and data packets (*DFSList*) generated from the source. In addition, two FIFO output queues were used to store the packets forwarded to the controller from the switch, and vice versa. Finally, two lists were used to record the *FlowID*s of the active TCP1 and TCP2 flows respectively. Similarly, two lists were maintained to record the *FlowID*s of the active UDP1 flows and active UDP2 flows. In both simulations, the lists were used to check whether the packets arriving at the switch belonged to a new flow or an existing flow.

In the SDN TCP simulations (Lai et al., 2018a), the source first sent a SYN packet to the switch. The SYN packet was redirected to the controller, which then forwarded the SYN packet to the switch and updated the flow table entry for the corresponding TCP flow (marking the beginning of the TCP flow lifetime). The SYN packet was forwarded to the destination later. Subsequent arriving packets belonging to the same TCP flow were similarly forwarded directly to the destination.

Table 3
Baseline parameters.

Parameter Name	Value
Data packet arrival rate per flow at switch, λ_P	250 packets/s
Flow arrival rate at switch, λ_F (λ_{TCP}, λ_{UDP})	150, 100 ∽200 flows/s
Service rate at switch, μ_S	50000 packets/s
Flow termination rate, μ_F (μ_{TCP}, μ_{UDP})	0.8, 0.4 ∽ 1.5 flows/s
Service rate at controller, μ_C	1000, 800 ∽ 1200 packets/s
Switch queue size, K_S	100 packets
Controller queue size, K_C	100 packets
Propagation delay, *DSC*	10 μs

In the SDN UDP simulations (Lai et al., 2018b), the source first sent a FIR packet to the switch, which redirected the packet to the controller (marking the beginning of the UDP flow lifetime). Data packets belonging to the same UDP flow continued to arrive at the switch while the FIR packet was handled by the controller, and were similarly redirected to the controller. The controller redirected the FIR packet to the switch and added a flow table entry for the UDP flow. The FIR packet was forwarded to the destination from the switch later. All subsequent data packets belonging to the same UDP flow were then forwarded to the destination directly without any further involvement of the controller.

In the TCP/UDP simulations (Lai et al., 2018a, 2018b), each SYN/FIR packet contained a unique *FlowID* and all of the data packets belonging to the same TCP/UDP flow contained the same *FlowID*. On arrival of a SYN/FIR packet (new flow) at the switch from the source, a SYN/FIR packet with *FlowID* was created and added to *SFSList*. An arrival event indicating the arrival of the SYN/FIR packet was then created with an updated event time and added to the *EPQ*. During the simulations, an event was polled from the *EPQ* and the corresponding event procedure was executed iteratively according to its type until the termination condition was reached.

For each simulation run, the simulation time was set as 3 s; with data being taken only after 1 s. For each measurement of interest, the simulation value was taken as the average value obtained over 10000 simulation runs. As stated above, the simulations were performed using a self-written JAVA simulator rather than with NS2 or NS3 since the TCP protocol used in NS2/NS3 considers the complete TCP functionality, i.e., connection establishment, packet retransmission, congestion control, and flow control, whereas the present models considered only the connection establishment functionality.

4.2. Parameter settings

Table 3 lists the baseline parameters used in the simulations and analytical models. To better interpret the results, the TCP and UDP flow arrival rate, i.e., λ_{TCP} and λ_{UDP}, respectively, were both denoted by a general flow arrival rate λ_F since both arrival rates have same value in the simulations and analytical models. Similarly, the TCP and UDP termination rates, μ_{TCP} and μ_{UDP}, were both denoted by a general flow termination rate μ_F.

Most of the baseline values in Table 3 are taken directly from Goto et al. (2016) since the study in Goto et al. (2016) developed a three-dimensional-state model, and is hence similar to the four-dimensional-state model adopted in the present study. The flow-level parameters, λ_P, λ_F, and μ_F, were chosen in such a way as to let the packet-level parameters and overall packet arrival rate be similar to the values set in Goto et al. (2016). Finally, in computing the propagation delay, *DSC*, the distance between the controller and the switch was assumed to be 2 km. Hence, the propagation delay was assumed to be 10 μs for a propagation speed of 2×10^8 m/sec.

The simulation and analytical results for the average packet delays and packet loss probabilities in the TCP and UDP models were compared for various values of the system parameters, λ_F, μ_F, and μ_C. To

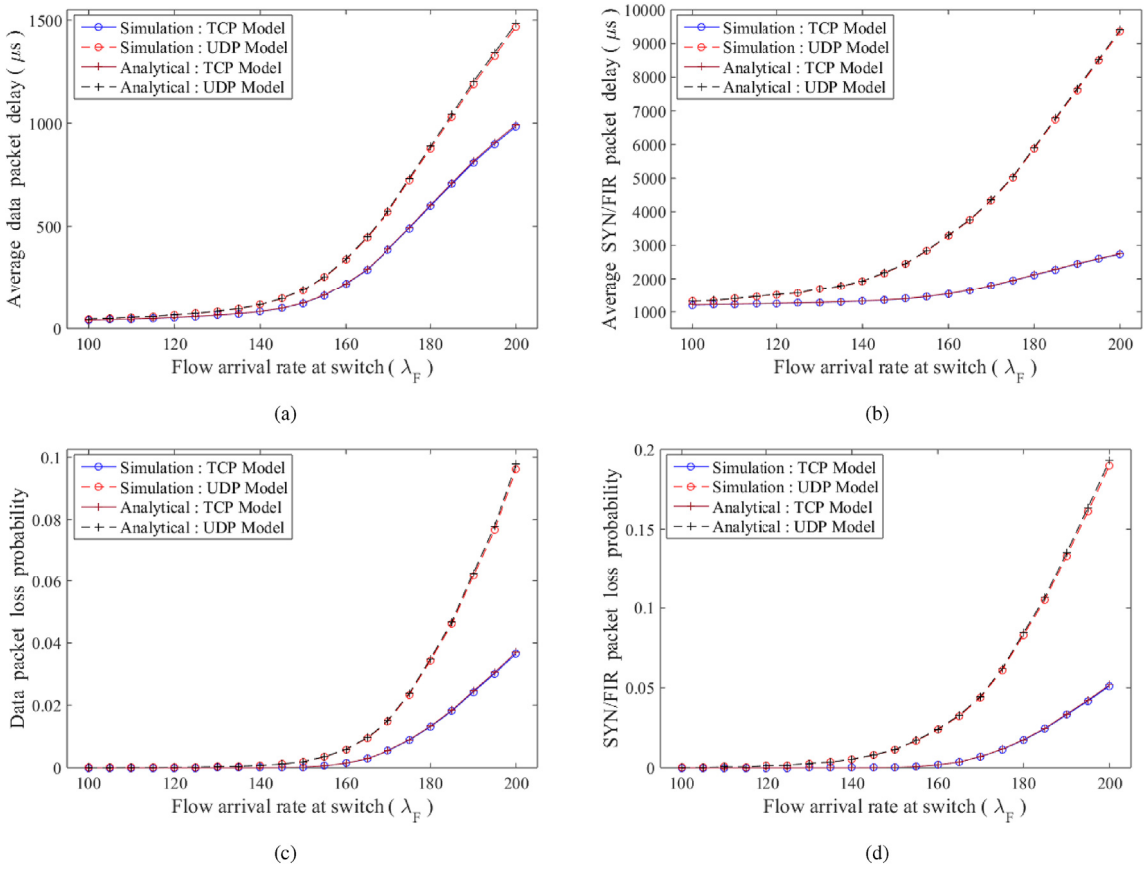

Fig. 6. Average packet delay and packet loss probability vs. λ_F.

observe the effects of each system parameter on the TCP and UDP performance, the parameters were selected in turn and assigned various values in a prescribed range while holding the other two parameters constant at their baseline values. In particular, the TCP and UDP flow arrival rate was varied from 100 to 200 flows/sec (with a baseline value of 150 flows/sec). Similarly, the TCP and UDP flow termination rate was varied from 0.4 to 1.5 flows/sec (with a baseline value of 0.8 flows/sec) while the controller service rate was varied from 800 to 1200 packets/sec (with a baseline value of 1000 packets/sec).

4.3. Impact of flow arrival rate, λ_F

Fig. 6(a) and (b) show the effects of λ_F on the average data packet delay, D^T_{data}, and average SYN packet delay, D^T_{SYN}, in the TCP model, and the average data packet delay D^U_{data}, and average FIR packet delay, D^U_{FIR}, in the UDP model. As shown in Fig. 6(a), D^T_{data} and D^U_{data} both increase exponentially with the increasing λ_F. Moreover, in Fig. 6(b), D^U_{FIR} increases exponentially, while D^T_{SYN} increases almost linearly. As λ_F increases, the number of new flows at the switch also increases; giving rise to a large data packet arrival rate and a greater number of SYN or FIR packets redirected to the controller (which adds further feedback to the switch). The larger number of data packets from the source, and the feedback packets from the controller, increase the number of packets required to wait in the switch queue. Consequently, the average switch queue length and packet delay both increase. The greater packet delay at switch results in a higher D^T_{data} and D^T_{SYN} in the TCP model and D^U_{data} and D^U_{FIR} in the UDP model.

For a constant λ_F, the data packet delay in UDP, i.e., D^U_{data} is much higher than that in TCP, i.e., D^T_{data}. This finding is reasonable since in the UDP model, a large number of data packets are redirected to the controller, and their delay D^U_{data} (shown in Eqn. (13)) thus includes both a delay at the controller (D^U_C) and a propagation delay (DSC) in addition to the packet delay at the switch (D^U_S). Conversely, in the TCP model, no data packets (other than the SYN packet) are redirected to the controller, and hence, D^T_{data} (shown in Eqn. (8)) contains only the packet delay at the switch (D^T_S). Observing Fig. 6(a), it is seen that the D^U_{data} curve for the UDP model is steeper than the D^T_{data} curve for the TCP model. This can be attributed to the greater rate of increase in the average switch queue length, $\overline{q^U_S}$, and average controller queue length, $\overline{q^U_C}$, in the UDP model, which can be further explained as follows:

- The controller incoming rate in the UDP model is higher than that in the TCP model due to the greater number of packets redirected by the switch. As a result, the average controller queue length, $\overline{q^U_C}$, increases rapidly, particularly at higher value of λ_F. By contrast, in the TCP model, $\overline{q^T_C}$ increases only relatively slowly with increasing λ_F since only the SYN packets are redirected to the controller.
- In the UDP model, data packets are fed back from the controller to the switch, and hence the average switch queue length, $\overline{q^U_S}$, increases. In the TCP model, such feedback of the data packets does not occur, and consequently $\overline{q^T_S}$ increases more slowly as λ_F increases.
- The number of active UDP flows is greater than the number of active TCP flows since the TCP flows are closed when the corresponding SYN packets are dropped, whereas in UDP, data packets continue to arrive at the switch even when the FIR packet is dropped. In other words, the data packet arrival rate at the switch in UDP is higher than that in TCP.

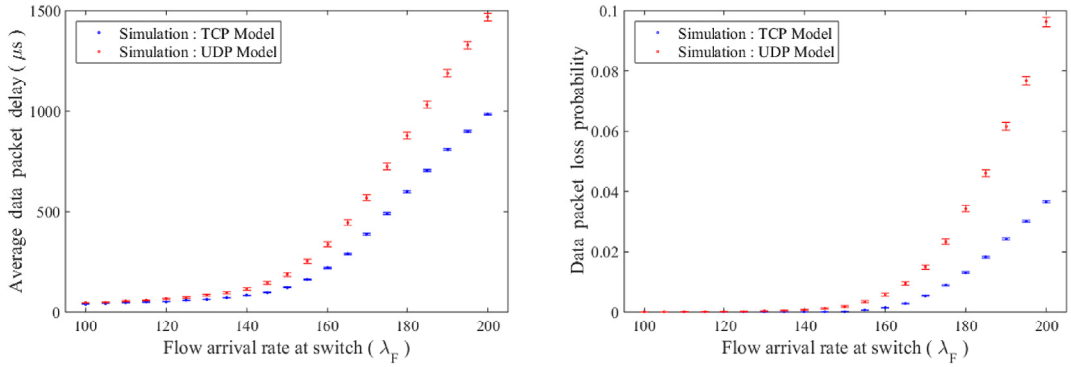

Fig. 7. Variations in average packet delays and packet loss probabilities in simulation runs.

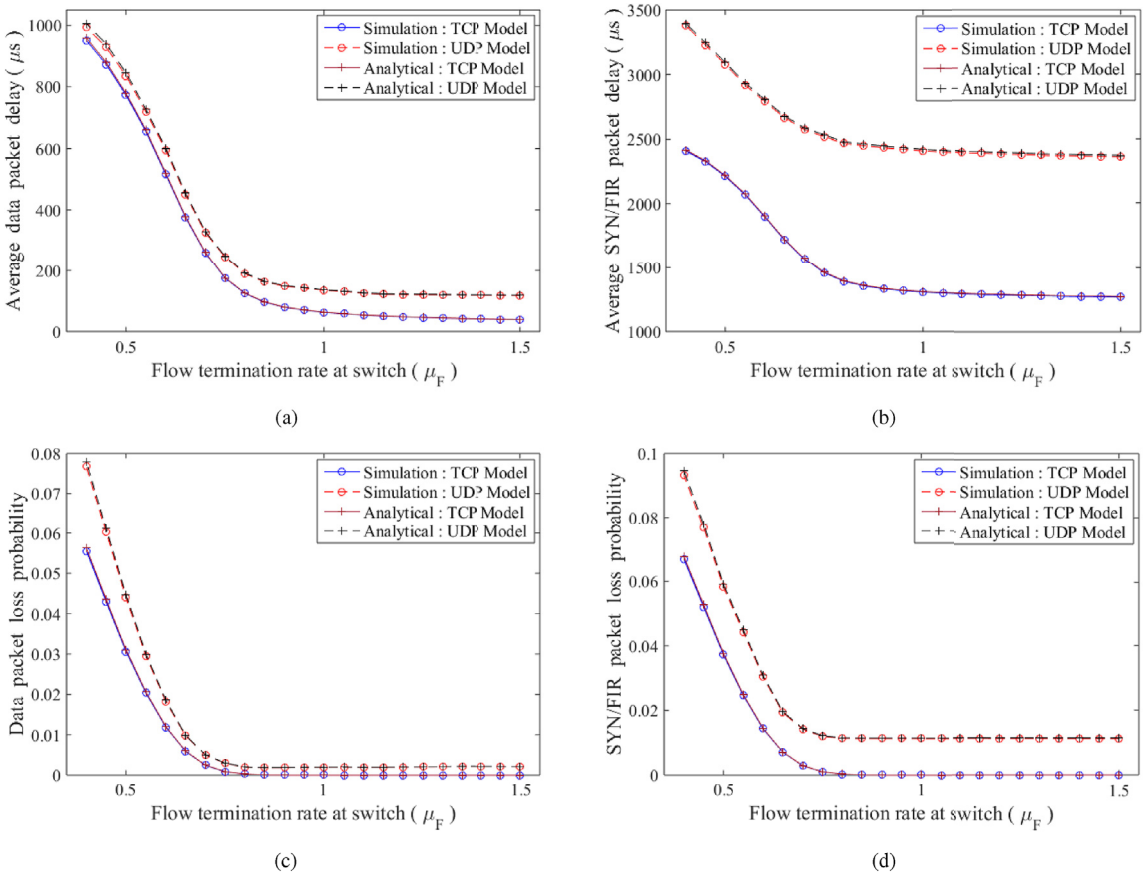

Fig. 8. Average packet delay and packet loss probability vs. μ_F.

- Finally, for each TCP flow, no data packets arrive at the switch before the corresponding SYN packet is handled by the controller. By contrast, for UDP flows, the data packets continue to arrive at the switch as the FIR packet is handled by the controller.

Fig. 6(c) and (d) show the impact of λ_F on the data packet loss probability, L^T_{data}, and SYN packet loss probability, L^T_{SYN}, in the TCP model, and the data packet loss probability, L^U_{data}, and FIR packet loss probability, L^T_{FIR}, in the UDP model. For both models, the loss probabilities increase exponentially with the increasing λ_F since, as described above, as λ_F increases, the average switch queue length also increases; causing a greater number of packets to be dropped at the switch. The controller and switch queue lengths increase more rapidly in the UDP model than in the TCP model. Consequently, the L^U_{data} and L^U_{FIR} curves in Fig. 6(c) and (d) are higher and steeper than the L^T_{data} and L^T_{SYN} curves, respectively.

Overall, the results presented in Fig. 6 show that, irrespective of the value of λ_F, the analytical results for the average packet delay and packet loss probability are in excellent agreement with the simulation

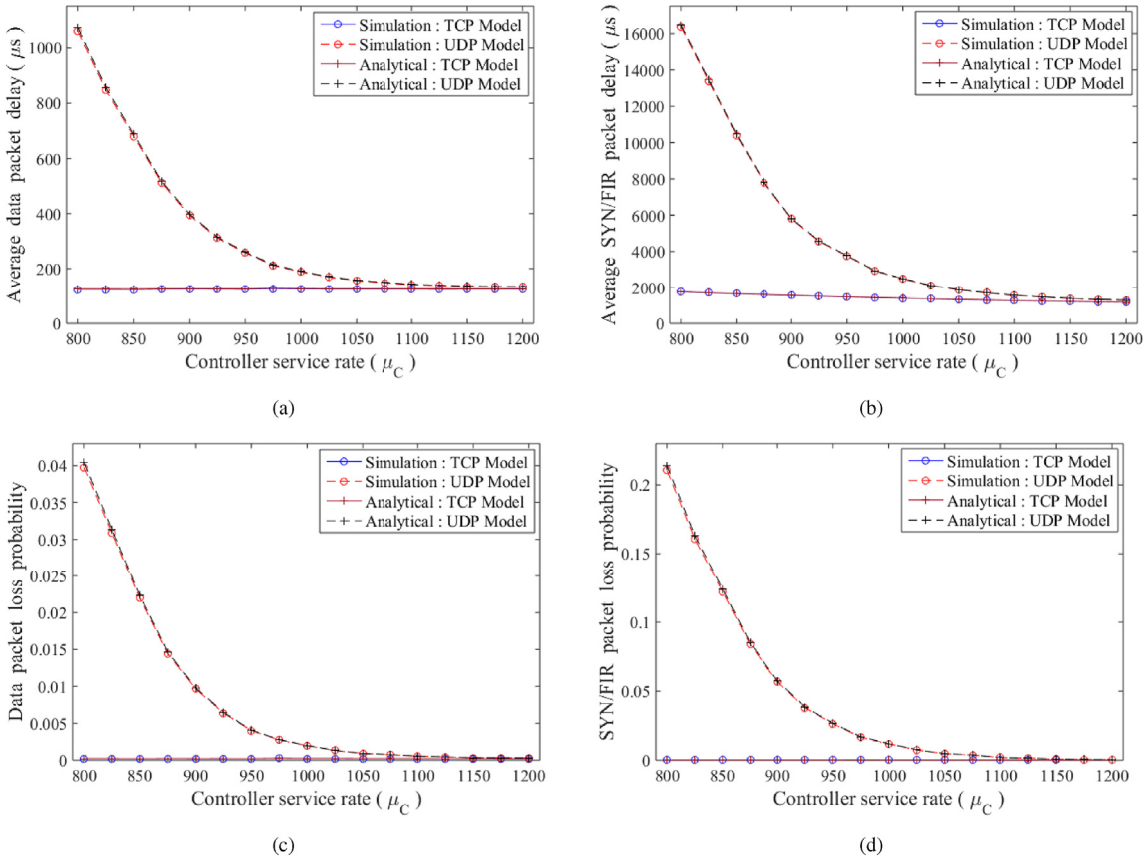

Fig. 9. Average packet delay and packet loss probability vs. μ_C.

results. Consequently, the validity of the proposed UDP and TCP SDN analytical models is confirmed.

As described above, the plotted simulation points were taken as the average values computed over 10000 simulation runs. Fig. 7(a) and (b) show the variation in the average packet delay and packet loss rate for each value of λ_F. Note that the results reflect a 99% confidence interval for D^T_{data} and L^T_{data} in the TCP model, and D^U_{data} and L^U_{data} in the UDP model. The small variation shows that the results are both stable and convincing.

4.4. Impact of flow termination rate, μ_F

Fig. 8 shows the effect of μ_F on the average packet delay and packet loss in the TCP and UDP models. As shown in Fig. 8(a) and (b), the packet delay decreases exponentially with increasing μ_F in both models. As μ_F increases, the lifetime of the TCP and UDP flows decreases. Consequently, the number of active flows and data packet arrivals at the switch reduces; leading to a reduction in both the average switch queue length and the packet delay at the switch. The smaller packet delay at the switch reduces D^T_{data} and D^T_{SYN} in the TCP model and D^U_{data} and D^U_{FIR} in the UDP model. Fig. 8(a) and (b) additionally show that the D^U_{data} and D^U_{FIR} curves for the UDP model are higher than the D^T_{data} and D^T_{SYN} curves for the TCP model. This can be attributed to the same reasons as those described above in Subsection 4.3 for the effect of the flow arrival rate, λ_F. Hence, the detailed explanation is omitted here for reasons of conciseness.

Fig. 8(c) and (d) show that the packet loss probability decreases exponentially with the increasing μ_F in both models. This result is reasonable since, as μ_F increases, the average switch queue length decreases, and hence the number of packets dropped at the switch reduces. Therefore, L^T_{data} and L^T_{SYN} in the TCP model, and L^U_{data} and L^U_{FIR} in the UDP model, all decrease.

In general, increasing λ_F and decreasing μ_F both result in a greater number of active flows in the system, and hence increase the packet arrival rate at the switch. Consequently, the packet delay at the switch increases in both cases. However, the packet arrival rate at the controller increases with increasing λ_F, but decreases with decreasing μ_F.

4.5. Impact of controller service rate, μ_C

Fig. 9 shows the effect of μ_C on the average packet delay and packet loss probability in the two models. It is seen in Fig. 9(a) that D^T_{data} remains approximately constant with increasing μ_C. By contrast, D^U_{data} decreases exponentially as μ_C increases. For a higher value of μ_C, the controller service time decreases, and hence fewer data packets are redirected to the controller in the UDP model before the FIR packet is handled by the controller. Consequently, the number of packets required to wait in the controller queue is also reduced; leading to a reduction in both the average controller queue length, $\overline{q^U_C}$, and the packet delay at the controller, D^U_C. The reduced number of feedback packets from the controller to the switch also decreases the average switch queue length, $\overline{q^U_S}$; thereby reducing D^U_{data} and D^U_{FIR}. By contrast, $\overline{q^T_S}$ and $\overline{q^T_C}$ in the TCP model remain approximately constant since the load is light, and hence D^T_{data} and D^T_{SYN} also remain almost constant.

Observing Fig. 9(c) and (d), it is seen that L^T_{data} and L^T_{SYN} remain near constant with increasing μ_C, whereas L^U_{data} and L^U_{FIR} decrease exponentially. As μ_C increases, $\overline{q^U_S}$, $\overline{q^U_C}$, and the number of incoming packets at

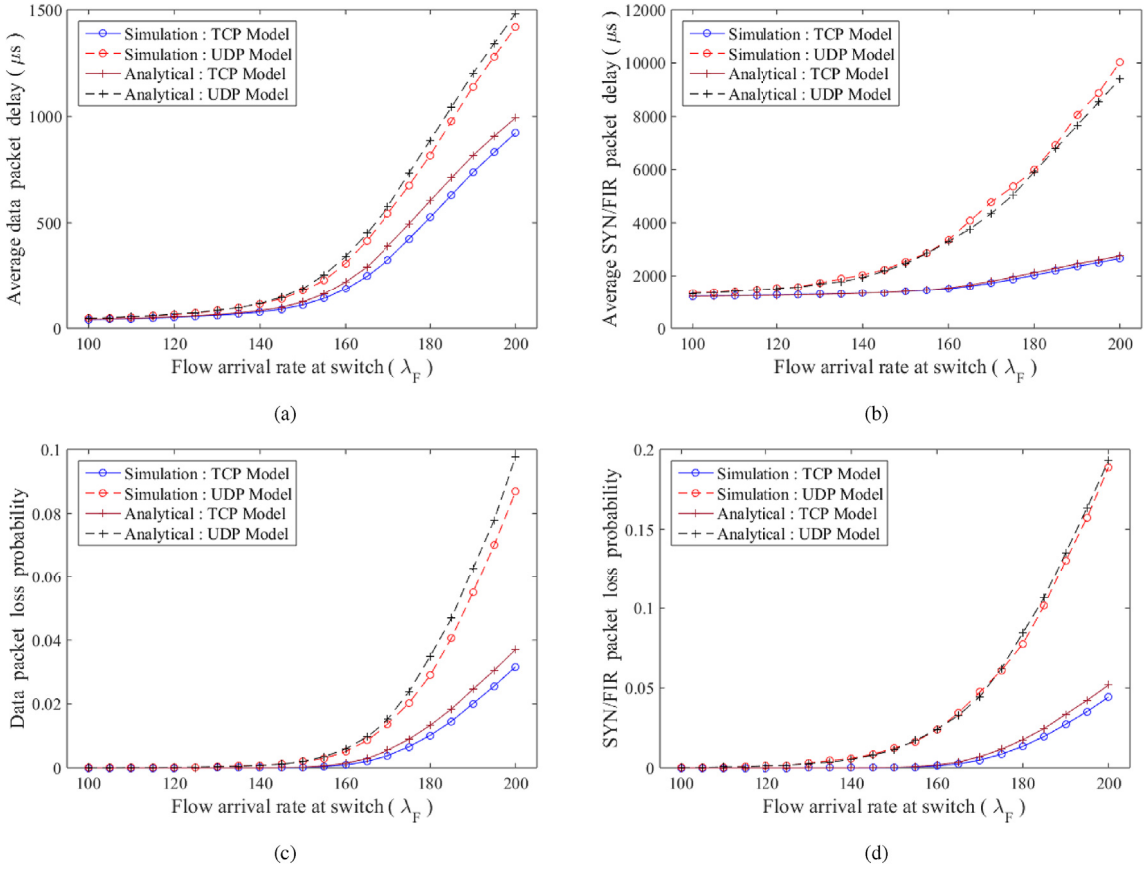

Fig. 10. Average packet delay and packet loss probability vs. λ_F under the flow inter-arrival time following Weibull distribution.

the controller all decrease; resulting in lower L^U_{data} and L^U_{FIR}. Conversely, as described above, $\overline{q^T_C}$ and $\overline{q^T_S}$ in the TCP model remain approximately constant for all values of μ_C, and hence L^T_{data} and L^T_{SYN} also remain almost constant.

4.6. Weibull distribution for flow inter-arrival time

In the results presented above, the TCP/UDP flow arrivals are assumed to follow a Poisson distribution. In other words, the flow inter-arrival time satisfies an exponential distribution. However, as decribed in Wu et al. (2007), Arshadi and Jahangir (2011) and Arfeen et al. (2013), the IP flow inter-arrival time in practical networks actually follows a Weibull distribution, which has a shape parameter close to 1.0 for large-scale networks. Hence, to further validate the present analytical models, the exponential distribution assumption for the flow inter-arrival time was replaced with a Weibull distribution. As in Wu et al. (2007) and Arshadi and Jahangir (2011), the shape parameter in the Weibull distribution was assigned a value of 0.95 and the baseline values were adopted for all the other parameters.

Fig. 10 shows the simulation and analytical results for the average packet delay and packet loss rate of the TCP and UDP models under various values of λ_F when using the Weibull distribution to model the flow inter-arrival time. As shown in Fig. 10(a) and (b), the average packet delay increases exponentially with increasing λ_F for both the simulation results and the analytical results. A detailed inspection shows that the analytical and simulation results for the TCP model deviate by around 9.92% for the data packet delay and is 1.92% for the SYN packet delay. Similarly, for the UDP model, the two sets of results deviate by 5.03% for the data packet delay and is 3.17% for the FIR packet delay.

Fig. 10(c) and (d) show the effect of λ_F on the packet loss probability in the TCP and UDP model. It is seen that for both models, the packet loss probabilities increase exponentially with increasing λ_F. Furthermore, a good agreement is once again observed between the simulation results and the analytical results in every case. For example, the analytical and simulation results for the data packet loss probability and SYN packet loss probability in the TCP model deviate by just 9.5% and 9.36%, respectively. Moreover, for the UDP model, the two sets of results deviate by just 6.8% for the data packet loss probability and 4.5% for the FIR packet loss probability. In general, the results presented in Fig. 10 confirm the feasibility of the TCP and UDP analytical models even when the flow inter-arrival time follows a more realistic Weibull distribution.

5. Conclusion

This study has developed two analytical models for evaluating the performance of TCP and UDP flows over SDN. For both models, the control plane and data plane have been considered jointly and the system has been represented as a four-dimensional state. The steady-state probabilities of both systems (i.e., TCP SDN and UDP SDN) have been computed and appropriate performance metrics derived. The validity of the models has been confirmed by means of extensive simulations.

The simulation results obtained for the TCP and UDP models have been compared for various performance metrics. It has been observed that D^T_{data} and D^U_{data} differ by around 12 ∽ 50%, D^T_{SYN} and D^U_{FIR} by 9 ∽ 72%, L^T_{data} and L^U_{data} by 25 ∽ 100%, and L^T_{SYN} and L^U_{FIR} by

28 ∽ 100%. In other words, the results for the packet loss probability under the two models differ more significantly than those for the packet delay. This result is reasonable since as the network becomes more congested, i.e., larger λ_F, lower μ_F, and lower μ_C, the differences between the two models becomes more pronounced. In particular, the performance of the UDP model is poorer than that of the TCP model mainly because a larger number of packets (i.e., FIR and some data packets) are redirected to the controller in the UDP model, whereas in the TCP model, only the SYN packets are routed to the controller.

The main aim of this study is simply to analyze and compare the performance of TCP and UDP flows over SDN using an analytical approach. To facilitate the analysis, several simplifying assumptions have been made regarding the two models. In addition, the full functionality of TCP has not been considered. Accordingly, future studies will perform NS2/3 simulations to further verify the present analysis and compare the performance of TCP and UDP flows over SDN under the full TCP functionality (i.e., connection establishment, retransmission, congestion control, and flow control).

References

Ali, A., 2018. Performance Analysis of TCP and UDP Flows over SDN Simulation Code. [Online]. Available: https://github.com/ahsan210/SDN.

Arfeen, M.A., Pawlikowski, K., McNickle, D., Willig, A., 2013. The role of the weibull distribution in internet traffic modeling. In: Teletraffic Congress (ITC), 2013 25th International. IEEE, pp. 1–8.

Arshadi, L., Jahangir, A.H., 2011. On the tcp flow inter-arrival times distribution. In: Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium on. IEEE, pp. 360–365.

Azodolmolky, S., Wieder, P., Yahyapour, R., Oct 10-11, 2013. Performance evaluation of a scalable software-defined networking deployment. In: 2013 Second European Workshop on Software Defined Networks (EWSDN). IEEE, Berlin, Germany, pp. 68–74.

Azodolmolky, S., Nejabati, R., Pazouki, M., Wieder, P., Yahyapour, R., Simeonidou, D., Dec 9-13, 2013. An analytical model for software defined networking: a network calculus-based approach. In: IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, pp. 1397–1402.

Barakat, C., Thiran, P., Iannaconne, G., Diot, C., Owerzazki, P., 2003. Modeling internet backbone traffic at the flow level. IEEE Trans. Signal Process. 51, 2111–2124 no. LCA-ARTICLE-2003-009.

Bari, M.F., Boutaba, R., Esteves, R., Granville, L.Z., Podlesny, M., Rabbani, M.G., Zhang, Q., Zhani, M.F., 2013. Data center network virtualization: a survey. IEEE Commun. Surv. Tutor. 15 (2), 909–928.

Bozakov, Z., Rizk, A., Oct 10-11, 2013. Taming sdn controllers in heterogeneous hardware environments. In: Second European Workshop on Software Defined Networks (EWSDN). IEEE, Berlin, Germany, pp. 50–55.

Drutskoy, D., Keller, E., Rexford, J., 2013. Scalable network virtualization in software-defined networks. Internet Comput. 17 (2), 20–27.

Fahmin, A., Lai, Y.-C., Hossain, M.S., Lin, Y.-D., 2018. Performance modeling and comparison of nfv integrated with sdn: under or aside? J. Netw. Comput. Appl. 113, 119–129.

Goto, Y., Masuyama, H., Ng, B., Seah, W.K., Takahashi, Y., September 19-21, 2016. "Queueing analysis of software defined network with realistic openflow–based switch model. In: IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), London, UK, pp. 301–306.

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T., December 2014. Software-defined networking: challenges and research opportunities for future internet. Comput. Network. 75 (A), 453–471.

Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N., April 28–30, 2010. Elastictree: saving energy in data center networks. In: 7th USENIX Conference on Networked Systems Design and Implementation. ACM, San Jose, California, pp. 1–17.

Holzle, U., 2012. Opening Address: 2012 Open Network Summit, vol. 8, p. 2014 no. 8 http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf Date Retrieved.

Jain, R., Paul, S., 2013. Network virtualization and software defined networking for cloud computing: a survey. IEEE Commun. Mag. 51 (11), 24–31.

Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P., 2011. Modeling and performance evaluation of an OpenFlow architecture. In: International Teletraffic Congress, San Francisco, California, USA, pp. 1–7.

Jie, Y., Li, W., Qiao, Y., Fadlullah, Z.M., Kato, N., 2015. Characterizing and modeling of large-scale traffic in mobile network. In: Wireless Communications and Networking Conference (WCNC). IEEE, pp. 801–806.

Lai, Y.-C., Ali, A., Hossain, M.S., Lin, Y.-D., April 2018. Performance Evaluation of TCP Connections over Software Defined Networks. Technical Report TR-BUET-18-01 [Online]. Available: http://teacher.buet.ac.bd/mshohrabhossain/publication.html, http://teacher.buet.ac.bd/mshohrabhossain/pub/TR-TCP-SDN-2018.pdf.

Lai, Y.-C., Ali, A., Hossain, M.S., Lin, Y.-D., April 2018. Performance Evaluation of UDP Flows over Software Defined Networks. Technical Report TR-BUET-18-02 [Online]. Available: http://teacher.buet.ac.bd/mshohrabhossain/publication.html, http://teacher.buet.ac.bd/mshohrabhossain/pub/TR-UDP-SDN-2018.pdf.

Mahmood, K., Chilwan, A., Østerbø, O.N., Jarschel, M., 2014. On the Modeling of Openflow-Based Sdns: the Single Node Case. arXiv:1411.4733.

Mahmood, K., Chilwan, A., Østerbø, O., Jarschel, M., 2015. Modelling of OpenFlow-based software-defined networks: the multiple node case. IET Netw. 4 (5), 278–284.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

Metter, C., Seufert, M., Wamser, F., Zinner, T., Tran-Gia, P., Oct 31–Nov 4, 2016. "Analytic model for sdn controller traffic and switch table occupancy. In: 12th International Conference on Network and Service Management. IEEE, Montreal, QC, Canada, pp. 109–117.

Metter, C., Seufert, M., Wamser, F., Zinner, T., Tran-Gia, P., 2017. Analytical model for sdn signaling traffic and flow table occupancy and its application for various types of traffic. IEEE Trans. Netw. Serv. Manag. 14 (3), 603–615.

Miao, W., Min, G., Wu, Y., Wang, H., Dec 19–21, 2015. Performance modelling of preemption-based packet scheduling for data plane in software defined networks. In: International Conference on Smart City/SocialCom/SustainCom (SmartCity). IEEE, Chengdu, China, pp. 60–65.

Miao, W., Min, G., Wu, Y., Wang, H., Hu, J., 2016. Performance modelling and analysis of software-defined networking under bursty multimedia traffic. ACM Trans. Multimed Comput. Commun. Appl. (TOMM) 12 (5s) pp. 77:1–19.

ONF, April 2012. Software-Defined Networking: the New Norm for Networks. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

ONF, December 2013. SDN Architecture Overview. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf.

Pan, J., Paul, S., Jain, R., 2011. A survey of the research on future internet architectures. IEEE Commun. Mag. 49 (7), 26–36.

Pries, R., Jarschel, M., Goll, S., 2012. On the usability of openflow in data center environments. In: IEEE International Conference on Communications (ICC), Ottawa, Canada, June 10-15, pp. 5533–5537.

Rowshanrad, S., Namvarasl, S., Abdi, V., Hajizadeh, M., Keshtgary, M., 2014. A survey on SDN, the future of networking. Adv. Comput. Sci. Technol. 3 (2), 232–248.

Wang, G., Li, J., Chang, X., June 15-16, 2015. "Modeling and performance analysis of the multiple controllers' approach in software defined networking. In: 23rd International Symposium on Quality of Service. IEEE, Portland, OR, USA, pp. 73–74.

Wu, H., Zhou, M., Gong, J., 2007. Investigation on the ip flow inter-arrival time in large-scale network. In: Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on. IEEE, pp. 1925–1928.

Xiong, B., Yang, K., Zhao, J., Li, W., Li, K., June 2016. Performance evaluation of OpenFlow-based software-defined networks based on queueing model. Comput. Network. 102, 172–185.

Yang, S.-R., Lin, P., Huang, P.-T., 2008. Modeling power saving for gan and umts interworking. IEEE Trans. Wireless Commun. 7 (12), 5326–5335.

Zhang, W., 2005. Performance of real-time and data traffic in heterogeneous overlay wireless networks. In: Proceedings of the 19th International Teletraffic Congress (ITC 19), Beijing, pp. 859–868.

Yuan-Cheng Lai received his Ph.D. degree in computer science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in 2001 and has been a professor since 2008. His research interests include wireless networks, network performance evaluation, network security, and social networks.

Ahsan Ali received his B.Sc. degree from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2017 in Computer Science and Engineering. He is working as Graduate Researcher at the department of CSE, BUET. His research interests include Wireless networks, Software defined networking and Network function Virtualization.

Md. Shohrab Hossain received his B.Sc. and M.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in the year 2003 and 2007, respectively. He obtained his Ph.D. degree from the School of Computer Science at the University of Oklahoma, Norman, OK, USA in December 2012. He is currently serving as an Associate Professor in the Department of CSE, BUET. His research interests include Mobile malware detections, cybersecurity, Software defined networking, security of mobile and ad hoc networks, and Internet of Things. He has been serving as the TPC member of IEEE Globecom, IEEE ICC, IEEE VTC, Wireless Personal Communication, (Springer), Journal of Network and Computer Applications (Elsevier), IEEE Wireless Communications. He serves as a member ofITEE Question Making Committee (QMC) for the ITPEC countries. He has published more than 50 technical research papers in leading Journals and conferences from IEEE, Elsevier, Springer.

YING-DAR LIN is a Distinguished Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose, California, during 2007–2008, and the CEO at Telecom Technology Center, Taipei, Taiwan, during 2010–2011. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic and has been an approved test lab of the Open Networking Foundation (ONF) since July 2014. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. His research interests include network security, wireless communications, and network cloudification. His work on multihop cellular was the first along this line, and has been cited over 800 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), and ONF Research Associate. He currently serves on the editorial boards of several IEEE journals and magazines, and is the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST). He published a textbook, Computer Networks: An Open Source Approach (www.mhhe.com/lin), with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).