Journal of Network and Computer Applications 122 (2018) 24–36

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Modelling Software-Defined Networking: Software and hardware switches

Deepak Singh [a,*], Bryan Ng [a,**], Yuan-Cheng Lai [b], Ying-Dar Lin [c], Winston K.G. Seah [a]

[a] *School of Engineering and Computer Science, Victoria University of Wellington, New Zealand*
[b] *Dept of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan*
[c] *Dept of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*

A R T I C L E I N F O

Keywords:
Modelling
Software-defined network
Software switch
Hardware switch
Queueing theory

A B S T R A C T

In Software-Defined Networking (SDN), a switch is a forwarding device that moves data packets in a network. A software switch handles forwarding functions in software and thus cannot forward packet at line speed while a hardware switch leverages optimised forwarding hardware to forward packets at line speed. However, there has been very little research in the literature to help network engineers understand the tradeoffs in choosing one over the other. In this paper, we develop a unified queueing model for characterizing the performance of hardware switches and software switches in SDN. The unified queueing model is an analytical tool for engineers to predict delay, packet loss and throughput in their SDN deployments. Existing queueing models of SDN have focused on performance analysis of software switches, while our work presented herein is the first to present a unified analysis of hardware and software switches. Our proposed models exhibit errors below 5% compared to simulation. Between a hardware and software switch, the evaluation shows that a hardware switch achieves an average 80% lower delay and up to 100% lower packet loss probability compared to a software switch. The more a hardware switch involves the controller for decisioning, the lower the gains in terms of packet delays through the switch.

1. Introduction

Software-Defined Networking (SDN) is an emerging network architecture that decouples the control plane from the data plane in the switch. With this decoupling, the control plane is logically centralized and has a network-wide view of packet forwarding that occurs in any given end-to-end path. The logically centralized control plane (also termed controller) programs the data plane via a “southbound interface” which is interface between controller and switch. OpenFlow is the de-facto protocol for southbound interface championed by the Open Networking Foundation (ONF) (ONF, 2014).

While the controller plays an important role in managing the network and handling control information, the work of moving data across the network is done by the switch, and therefore the performance of SDN is a function of the switch forwarding speeds and the switch-controller interaction. Analytical models of SDN switches are a key step for SDN performance characterization and these models will help network engineers design networks suitable for delay and packet loss sensitive applications like industrial automation system (Haiyan et al., 2016), interactive video (Nayak et al., 2016), online surgery (Kumar et al., 2017) etc.

In this paper, we are concerned with physical SDN switches as opposed to virtual switches such as those instantiated in virtual machines. Within such a context, SDN switches are generally categorized into software switches and hardware switches. A software switch maintains the flow table in SDRAM (synchronous dynamic random access memory) where the incoming packet is matched against the flow table entries (FTE) using a CPU (central processing unit). If there is no matching FTE, packet is forwarded to the controller which feedback forwarding information to the switch and update the software flow table. The packet processing logic in a software switch is implemented in software (Goransson and Black, 2014) usually with the help of optimised software libraries. Open vSwitch (OVS) (Open vSwitch), Pantou/OpenWRT (Pantou), ofsoftswitch13 (ofsoftswitch13), Indigo (Indigo: Open Source Openf) running on commodity hardware (e.g. desktops with several network interface cards) are few examples

* Corresponding author.
** Corresponding author.
E-mail addresses: deepak.singh@ecs.vuw.ac.nz (D. Singh), bryan.ng@ecs.vuw.ac.nz (B. Ng), laiyc@cs.ntust.edu.tw (Y.-C. Lai), ydlin@cs.nctu.edu.tw (Y.-D. Lin), winston.seah@ecs.vuw.ac.nz (W.K.G. Seah).

https://doi.org/10.1016/j.jnca.2018.08.005
Received 11 March 2018; Received in revised form 4 June 2018; Accepted 14 August 2018
Available online 17 August 2018

of SDN software switches.

Similarly, in a SDN hardware switch, packet processing function is embedded in specialized hardware. This specialized hardware includes layer two forwarding tables implemented using content-addressable memories (CAMs), layer three forwarding tables using ternary content-addressable memories (TCAMs) (Goransson and Black, 2014) and application specific integrate circuits (ASICs). In a hardware switch, the FTEs is stored in CAMs and TCAMs of specialized hardware and packets are processed by the ASICs. Hardware switches are also equipped with SDRAM and CPU allowing a hardware switch to maintain flow tables in both TCAM and SDRAM (Rygielski et al.).

For hardware switches, line speed packet matching is possible because of the CAM and TCAM. However, most switches have limited TCAM capacity to store FTE due to the cost, size and energy consumed by TCAM (Rawat and Reddy, 2017). On the other hand, SDRAM is cheaper, consumes less power for storing FTE besides offering more flexibility to implement complex actions through software. Hence, much more SDRAM (compared to TCAM) is available in a switch. Switches such as the Mellanox SN2000 series, NoviFlow NoviSwitch class of switches, HP ProCurve J9451A, Fulcrum Monaco Reference, Quanta LB4G, and Juniper Juno MX-Series are classified as hardware switches (Nunes et al., 2014; Huang et al., 2013).

Both software and hardware switches have strengths and weaknesses, and the choice of switch directly affects the performance of the network. To identify the potential bottlenecks that could hinder the performance of the SDN, the trade offs between choosing a hardware vs. software need to be studied and investigated to improve the performance of SDN. While other approaches like Network Calculus has been also used to model SDN (Azodolmolky et al., 2013; Koohanestani et al., 2017; Huang et al., 2017), we use queueing theory to model both hardware and software switches in this paper for finer grain analysis. Most analytical models (queueing and network calculus) for SDN characterize the software switch architecture (i.e. a switch equipped with CPU only), with none of the queuing models to the best of our survey have modelled the hardware switch architecture (i.e. switch with CPU and specialized hardware).

In this paper we have two objectives, (i) develop an analytical model for SDN hardware switch, and (ii) characterize the performance of SDN software and hardware switches. Analytical modelling of the SDN hardware switch provides important insights for benchmarking switch performance and parametric sensitivity analysis to help network engineers identify critical factors that may influence network performance. Finally, by comparing software and hardware switch, we develop guidelines to guide deployment choices such as under what operating conditions, software data plane outperforms hardware data plane and vice versa.

The rest of the paper is organized as follows. We explain the models for software and hardware SDN switch in Section 2. In Section 3, we discuss the related works on SDN performance analysis using queueing theory. In Section 4, we describe the queueing models for software and hardware SDN switches using a Quasi-Birth Death (QBD) process. In Section 5, we describe queueing model for software SDN switch in detail. In Section 6, we describe our proposed queueing model for hardware SDN switch. The results of our proposed queueing model for SDN hardware switch and comparison with SDN software switch is discussed in Section 7. Finally, we conclude the paper with discussion and future work in Section 8.

2. Packet flow in software and hardware SDN switches

A generic block diagram of a software switch where the external packet arrives at the switch and the switch is connected to a controller is shown in Fig. 1. There are three important phases an SDN model with software switch must capture. Phase (1), the first packet of a flow arrives at the switch and there is no matching FTE for the packet in SDRAM. Phase (2), the packet without a matching flow entry is forwarded to the controller **or** a packet with the matching FTE is serviced by the switch and forwarded to the destination. All packet processing and forwarding in the switch is executed on CPU and SDRAM. Finally, Phase (3), the controller feeds the forwarding information back to the switch and updates the flow table in the switch.

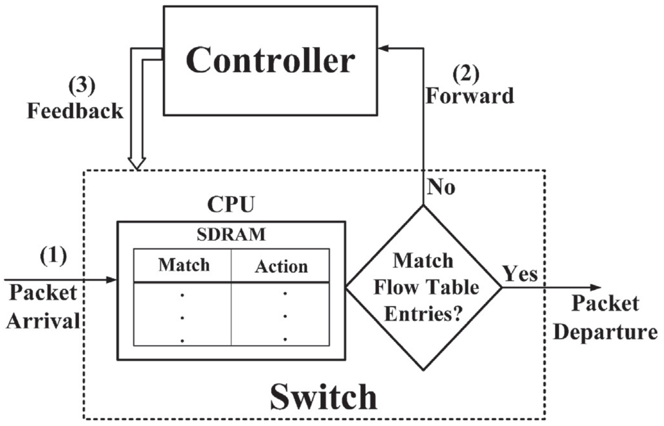


Fig. 1. Generic model for SDN with Software Switch.

Fig. 2 shows the block diagram of a hardware switch where the switch maintains flow tables in both hardware and software. The hardware and software flow tables are synchronized through a middleware layer on the switch (Kuźniar et al., 2015; Pan et al., 2013) to avoid duplicate entries and to ensure consistent forwarding behavior.

There are four important phases an SDN model with hardware switch must capture. Phase (1), the first packet of a flow arrives at the specialized hardware in switch that maintain hardware flow table entries and there is no matching FTE for the packet. Phase (2), a packet with the matching FTE in the TCAM is serviced by the ASIC and forwarded to the destination, otherwise a packet without a matching FTE in TCAM is matched against the FTE in SDRAM and processed by the CPU for forwarding to the destination. In phase (3), a packet without any matching entry in TCAM or SDRAM is forwarded to the

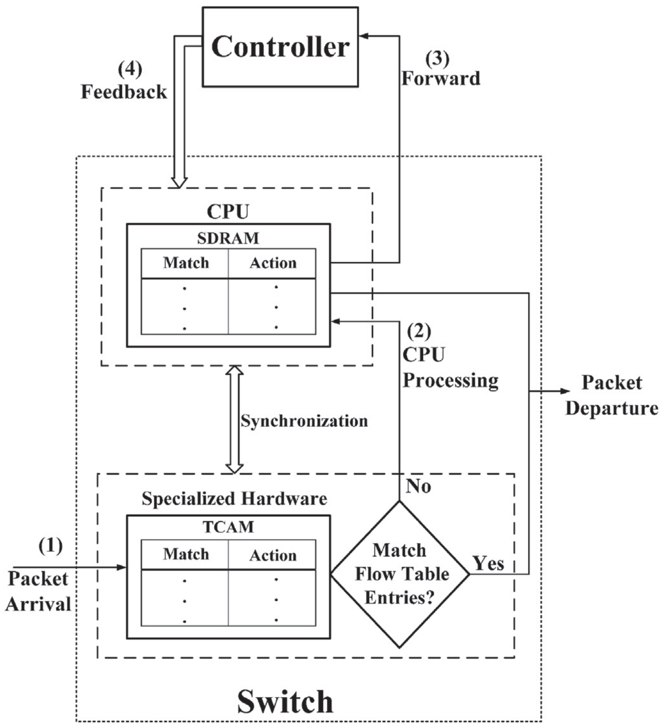


Fig. 2. Generic model for SDN with Hardware Switch.

controller. In phase (4), the controller feeds the forwarding information back to the switch, updates the flow tables in both TCAM and SDRAM. Finally, the packet is serviced by the CPU and forwarded to the destination.

3. Related work

Most of the previous works have modeled a software switch where the output buffers of the switch block in Fig. 1 is modeled either as a: (i) single shared queue or (ii) a two-priority queue (fast path vs. slow path). In the shared queue model, packets passing through the fast path and slow path share a single queue with first in first out (FIFO) service discipline while in the two-priority queue fast path and slow path packets are queued separately and each queue is served in a FIFO manner without preemption. In the two-priority queue model, packets in the slow path are always served with higher priority over fast path packets but the server does not preempt service of fast path packets once it has commenced, this is referred to as service without preemption.

The analytical model in (Jarschel et al., 2011) models the switch as a shared M/M/1 queue and the controller block modelled by an M/M/1/K queueing system respectively. In this model, the switch queue is a software switch and a fraction of incoming external traffic to the switch is forwarded to the controller. This assumption of independence (between switch and controller) together with the infinite buffer assumption preserves the Markovian property of the individual queues. Both assumptions help reduce the complexity in the resulting queueing network and yields a model that is amenable to product-form analysis. Product-form analysis decomposes the stationary probability distribution of the queueing network into the product of marginal probability distribution of each queue (Burke's theorem (Burke, 1956; Busic et al., 2012)), thus greatly simplifying analysis. While the shared queue model simplifies analyses, it does not reflect the realities of the different packet processing time scales of the hardware switch vs. the software switch in that it has no provisions for an accommodating the differentiation between the processing speeds of hardware and software switches.

The model presented in (Mahmood et al., 2014) considers switch and controller collectively as Jackson's network mimicking the model in (Jarschel et al., 2011). The model in (Mahmood et al., 2014) differs from (Jarschel et al., 2011) in that it does not assume that the controller and switch are independent. The traffic forwarded by the controller to the switch mixes with the external packet arrivals to the switch and this mixing better reflects the operational realities of an SDN switch. However, a single server in the model in (Mahmood et al., 2014) does not reflect the hardware processing speeds and internal queueing structure of hardware switches. To model a hardware switch through the model in (Mahmood et al., 2014) will necessitate a multi-server queue in place of a single server.

The model presented in (Yen and Su, 2014) have modelled SDN-based cloud computing as a two-stage tandem network. In this work, they have assumed switch as M/M/1 model with simple approximate analysis and does not distinguish the control and data traffic. The model presented in (Shang and Wolter, 1608) has assumed that the switch service time has a two-phase hyperexponential distribution and therefore modelled as an $M/H_2/1$ queue which was studied earlier in (Xiong et al., 2016). In the work of (Shang and Wolter, 1608), there are two different service distribution for packets arriving at the switch, one that does not have matching information and needs to be forwarded and other which has matching information. While both analyses do not distinguish between hardware and software switches, the model in (Shang and Wolter, 1608) does pave the way for modelling hardware switches via the two phases of the hyperexponential distribution.

The Log-Normal Mix Model (LNMM) has been proposed for OpenFlow switch in (Javed et al., 2017) to determine the path latency. In this model, they have assumed switch as M/G/1 model with log-normal mixture model as the service distribution. They have further demonstrated M/M/1 as poor fit for OpenFlow switch through experiments performed on Mininet, MikroTik Routerboard 750GL and GENI but did not consider it from the perspective of hardware and software switch. The lack of separation between the control plane and data plane packets in the model by (Javed et al., 2017) makes it a poorer fit (compared to other published works) for modelling SDN switches.

The model presented in (Miao et al., 2015) uses preemptive priority queues in switch with infinite capacity. This work also compares priority queueing buffer and shared buffer with simpler analysis concluding priority queueing as better buffer sharing mechanism. This work was extended into (Miao et al., 2016) which assumes arrival as Markov Modulated Poisson Process under bursty multimedia traffic scenario, and have assumed infinite capacity of high priority queue and finite capacity for low priority queue. Both the work in (Javed et al., 2017) and (Miao et al., 2015) have features suitable for modelling hardware switches because it can accommodate the different processing speeds in hardware and software. The processing speeds for hardware and software were characterized by different distributions of service times.

To reflect a realistic OpenFlow switch more accurately, priority queues with finite capacity are used instead (Goto et al., 2016). The priority queue in this model is non-preemptive whereby the lower priority queue is serviced when there are no packets in the higher priority queue. The reality of limited buffer sizes is then specifically addressed by a C-M/M/1/K/∞ queueing model that has been proposed for computing the minimum buffer size requirement of an OpenFlow switch (Mondal et al., 2018). The model presented in (Sood et al., 2016) is among the first to model SDN hardware switch using queueing theory but does not consider a hardware data plane. In this work, switch is assumed as M/Geo/1 model where inter-arrival is exponentially distributed and the service time is geometrically distributed but have not accounted the switch-controller interaction. In this work, the performance of the switch is defined as the time required by the switch to process the packets without any interaction with controller. However, the analysis in this work does not map the workings of hardware switch such as the flow matching and dedicated packet processing to the queueing model.

In (Singh et al., 2017), we showed that a priority queue used in (Goto et al., 2016) is closer to the realities of SDN compared to a shared buffer model in that it predicts the average time to install FTE and packet loss more closely to simulation. In this paper, we build upon that finding and devise a new model for analyzing both software and hardware switches.

Based on our survey of related research presented in this section, we summarize our findings in Table 1. In Table 1, the first column denotes the type of queue used (either a single shared buffer or a priority queue), the second column denotes the queueing model in Kendall's notation while the third and fourth column denote the type of analysis (exact vs. approximation) and the switch type (hardware vs. software). It is clear that apart from (Sood et al., 2016) we have not found other works that model SDN hardware switch. In the following sections, we describe the process of modelling SDN switches with queueing theory and show that a queueing model for SDN software and hardware switches is suitably captured by a QBD process.

4. Queueing models for software and hardware SDN switches

We develop two queueing models to study the effect of types of data plane on switch performance that is software or hardware data plane. These models called Model SPE (as used in (Goto et al., 2016)) and Model HPE are based on different data plane types in the switch. In Models SPE and HPE, "S" refers to the software switch, "H" refers to

Table 1
Summary of queueing models for SDN switches.

Model	Queue		Switch model	Analysis		Switch Type	
	Shared Buffer	Priority Queue		Exact	Approximate	Software	Hardware
Jarschel (Jarschel et al., 2011)	✓		M/M/1		✓	✓	
Mahmood (Mahmood et al., 2014)	✓		M/M/1		✓	✓	
Yen (Yen and Su, 2014)	✓		M/M/1		✓	✓	
Miao (Miao et al., 2015)	✓	✓	M/M/1		✓	✓	
Shang (Shang and Wolter, 1608)	✓		$M/H_2/1$		✓	✓	
Sood (Sood et al., 2016)	✓		M/Geo/1		✓		✓
Miao (Miao et al., 2016)		✓	MMAP		✓	✓	
Goto (Goto et al., 2016)		✓	GI/M/1/K	✓		✓	
Javed (Javed et al., 2017)	✓		M/G/1		✓	✓	
Singh (Singh et al., 2017)	✓	✓	GI/M/1/K	✓		✓	
Our Analysis		✓	GI/M/1/K	✓		✓	✓

the hardware switch,"E" refers to the encapsulation method in switch where data packets are encapsulated with control information and forwarded to the controller, and "P" refers to the priority queue in data plane.

Recall from earlier discussion in Section I that the software switch runs forwarding software (e.g. OpenVswitch, Lagopus or DPDK) on CPU rather than hardware microcoded forwarding engines, thus a single server model is appropriate. For brevity, we denote Model SPE as SPE and Model HPE as HPE.

Throughout this paper, we assume that the controller has an infinite capacity queue with M/M/1 distribution, the CPU (central processing unit) and the switch hardware both has a finite capacity with GI/M/1/K and M/M/1/K distributions, respectively. The external arrival at the switch is assumed to be Poisson and is denoted by λ_1, the service rate of the switch in SPE is denoted by μ_s, the service rate of the CPU in HPE is denoted by μ_{sp}, the service rate of the switch hardware in HPE is denoted by μ_{sh}, and the service rate of the controller is denoted by μ_c. If an incoming data packet to the switch hardware has no matching FTE (i.e. table miss), the packet is processed by CPU and forwarded to the controller, and this occurs with probability referred as table miss probability represented as β.

The average packet transfer delay is the primary performance metric used for comparing Model SPE and Model HPE which is commonly denoted by mean sojourn time of packet in queueing theory. Also, when using Little's theorem (i.e. mean sojourn time of packet is ratio of average packet length and arrival rate), the arrival rate to the controller is the throughput for the finite buffer (denoted by T) in the switch due to the packet losses.

4.1. QBD process for modelling SDN switches

The modelling approach in this paper is based on QBD processes, hence we devote this subsection to describing the notation and concepts behind QBD processes. A QBD process is continuous time Markov process which has an infinite number of levels with finite number of phases. A queueing network is represented by QBD process by mapping the number of packets in the network to levels and phases in the queueing model, i.e.

$$Queueing\ Network = \{Level, Phase\}.$$

For the SDN switch, the level variable tracks the number of packets in the controller and the phase variable tracks the number packets in the switch. The exact distribution probabilities of QBD process is determined using matrix geometric solution. These distribution probabilities can be used to determine various performance metrics of the queueing network like average delay, throughput.

Using standard QBD notation (Neuts, 1994), the transition rate matrix of QBD process is given by infinitesimal generator matrix Q with a repetitive block structure

$$Q = \begin{pmatrix} B_1 & A_0 & 0 & \dots & \\ A_2 & A_1 & A_0 & \ddots & \\ 0 & A_2 & A_1 & A_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

where, $A_0, A_1, A_2,$and B_1 are standard notation for sub-matrices that represent the phase distribution. The sub-matrices B_1 and A_1 represent phase distribution when level variable remains unchanged. For SDN network, the sub-matrix B_1 represents the boundary state of the network when the level variable is "0" (i.e. controller has no packets in queue), while A_1 represents the network with the controller having at least one packet in its queue. Similarly, A_0 represents the phase distribution when level variable increases by 1 (i.e. number of packets in the controller increases by 1), and A_2 represent phase distribution when level variable decreases by 1 (i.e. number of packets in the controller decreases by 1).

The exact distribution probabilities or the stationary state distribution (π) for QBD process is obtained by solving the equation, $\pi Q = 0$ and $\pi e = 0$, where e is the column vector with all elements as one. The QBD process is positive recurrent and irreducible if it satisfies the condition of, $\pi_A(A_0 - A_2) < 0$, where π_A is the stationary state distribution of $A_0 + A_1 + A_2$.

In the matrix geometric approach, only one queue or node is allowed to have an infinite capacity in order to avoid sub-matrices to be finite (Takagi, 1990). Because we are characterizing switch performance, we use the phase variable to track the switch and the level variable to track the SDN controller.

5. Model SPE: SDN software switch with priority queueing

Fig. 3 shows the model that uses priority queues for the software switch with finite capacity. The priority queues provides isolation between the packets arriving from the controller and external packets arriving at the switch. In this model, non-preemptive priority queues are used in which lower priority queue are serviced when there are no packets in the higher priority queue. The two priority classes proposed in (Goto et al., 2016) are called Class CS (controller to switch), and Class ES (external arrival to switch) to indicate the different packet processing paths. The control packets feedback by the controller is classified as Class CS packets and has higher priority over ES packets. Whereas, any packets other than control packets arriving at the switch has lower priority and is classified as Class ES packets.

Class ES represents the class for external data packet arrival to the switch. If the external data packet has no matching entry in the switch, the packet is sent to the controller, and this occurs with probability β. Class CS represents the class for packets fed back to the switch from the controller and must be forwarded out to the destination. Packets in the ES already have a forwarding rule installed in the switch and are

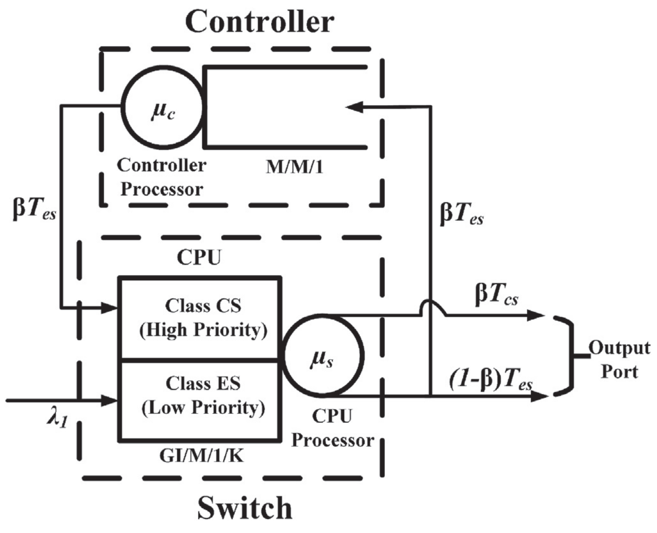


Fig. 3. Model SPE – a single server processing packets in the switch.

assigned to a lower priority with respect to Class CS packets. Both Class CS and Class ES queues shares service rate μ_s.

The SPE is modelled as a continuous time Markov chain $\{(n_c(t), n_{cs}(t), n_{es}(t)), t \geq 0\}$ on the state space $\{(x, y, z); x \in \mathbb{Z}_+, y \in \mathbb{Z}_+^{\leq K_1}, z \in \mathbb{Z}_+^{\leq K_2}\}$, where $n_c(t)$, $n_{cs}(t)$ and $n_{es}(t)$ are the number of packets in controller, class CS, and class ES, respectively at time t. At time t, we say that the processes $\{(n_c(t), n_{cs}(t), n_{es}(t))\}$ take on random values $\{x, y, z\}$ which yields the following:

$$\{n_c(t), n_{cs}(t), n_{es}(t)\} = \{x, y, z\}. \tag{1}$$

The queue capacity for switch is equal to K. It is assumed that Class CS and Class ES has the queue capacity of K_1 and K_2 respectively with the total queue capacity of switch as K.

The permissible transitions for the Markov chain $\{(n_c(t), n_{cs}(t), n_{es}(t))\}$ are shown in Table 2 and these help us derive the sub-matrices (that are A_0, A_1, B_1 and A_2) of transition rate generator matrix determine (Q) for Model SPE. These sub-matrices are input to the matrix geometric solution to compute the stationary distribution probability (π) which is used to determine the performance metrics for SPE.

In the following four subsections, we specify the expressions for the elements of the sub-matrices A_0, A_1, B_1 and A_2 using standard QBD notation (Neuts, 1994).

5.1. Elements of matrix A_0

The sub-matrix A_0 represents the phase distribution of Class CS and Class ES when the number of packets in controller (i.e. $n_c(t)$ or x in Eq. (1)) increases by 1.

$$A_{0(y,y')} = \begin{cases} A_{01}{}^{(y)}, & y' = y = 0, \\ 0, & \text{otherwise,} \end{cases}$$

where,

$$A_{01}{}^{(0)}_{(z,z')} = \begin{cases} \mu_s \beta, & z' = z - 1, \\ 0, & \text{otherwise.} \end{cases}$$

5.2. Elements of matrix A_1

The sub-matrix A_1 represents the phase distribution of Class CS and Class ES when the number of packets in controller remain unchanged and there are some packets in the controller (i.e. $n_c(t)$ or x in Eq. (1) is a positive integer that remains unchanged).

$$A_{1(y,y')} = \begin{cases} A_{11}{}^{(y)}, & y' = y, \\ A_{12}{}^{(y)}, & y' = y - 1, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$A_{11}{}^{(y)}_{(z,z')} = \begin{cases} \lambda_1, & z' = z + 1, \\ \mu_s(1-\beta), & y = 0, z' = z - 1, \\ -\lambda_1 - \mu_c, & y = 0, z = z' = 0, \\ -\lambda_1 - \mu_c - \mu_s, & y = 0, 0 < (z = z') < K_2, \\ -\lambda_1 - \mu_c - \mu_s, & 0 < y \leq K_1, 0 \leq (z = z') < K_2, \\ -\mu_s, & 0 \leq y \leq K_1, z = z' = K_2, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$A_{12}{}^{(y)}_{(z,z')} = \begin{cases} \mu_s, & z' = z, \\ 0, & \text{otherwise.} \end{cases}$$

5.3. Elements of matrix A_2

The sub-matrix A_2 represents the phase distribution of Class CS and Class ES when the number of packets in controller (i.e. $n_c(t)$ or x in Eq. (1)) decreases by 1.

$$A_{2(y,y')} = \begin{cases} A_{20}{}^{(y)}, & y' = y + 1, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$A_{20}{}^{(y)}_{(z,z')} = \begin{cases} \mu_c, & z' = z, \\ 0, & \text{otherwise.} \end{cases}$$

5.4. Elements of matrix B_1

The sub-matrix B_1 is identical to A_1 that represents the phase distributions of Class CS and Class ES when the number of packets in controller (i.e. $n_c(t)$ or x in Eq. (1) is equal to 0).

$\therefore B_1 = A_1(\text{for } \mu_c = 0)$.

Table 2
Permissible transitions for model SPE.

Event	From	To	Rate
One packet departs from Switch to out of system (SPE)	$(n_c, 0, n_{es})$	$(n_c, 0, n_{es} - 1)$	$\mu_s(1-\beta)$
One packet departs from Class CS to out of system (SPE)	$(n_c, n_{cs} > 0, n_{es})$	$(n_c, n_{cs} - 1, n_{es})$	μ_s
One packet arrives to Class ES	(n_c, n_{cs}, n_{es})	$(n_c, n_{cs}, n_{es} + 1)$	λ_1
One packet forwarded from Class ES to Controller	$(n_c, 0, n_{es})$	$(n_c + 1, 0, n_{es} - 1)$	$\mu_s\beta$
One packet serviced by Controller to Class CS	(n_c, n_{cs}, n_{es})	$(n_c - 1, n_{cs} + 1, n_{es})$	μ_c

5.5. Performance metrics

With these sub-block matrices, we can compute the stationary distribution probabilities ($\pi_{i,j,k}$) for i packets in controller, j packets in Class CS, and k packets in Class ES using the matrix geometric approach. The performance metrics like throughput of the controller and switch, the average time to install FTE in the switch, average packet loss in the switch can be computed using ($\pi_{i,j,k}$).

The throughput of controller (T_c) for Model SPE is given by the sum of probabilities that the controller has at least one packet to forward to Class CS with service rate of μ_c and this is given by:

$$T_c = \mu_c \sum_{i=1}^{\infty} \sum_{j=0}^{K_1} \sum_{k=0}^{K_2} \pi_{i,j,k}. \tag{2}$$

Similarly, the throughput of Class CS (T_{cs}) is given by the sum of probabilities that the Class CS has at least one packet to forward with service rate of μ_s and this is given by:

$$T_{cs} = \mu_s \sum_{i=0}^{\infty} \sum_{j=1}^{K_1} \sum_{k=0}^{K_2} \pi_{i,j,k}. \tag{3}$$

Also, the throughput of Class ES (T_{es}) is given by the sum of probabilities that the Class ES has at least one packet to forward with service rate of μ_s and no packet in Class CS in the stationary state, and this is given by

$$T_{es} = \mu_s \sum_{i=0}^{\infty} \sum_{k=1}^{K_2} \pi_{i,0,k}. \tag{4}$$

The average number of total packets for Model SPE is given as,

$$E[L]_{SPE} = \sum_{i=0}^{\infty} \sum_{j=0}^{K_1} \sum_{k=0}^{K_2} (i+j+k)\pi_{i,j,k}. \tag{5}$$

Applying Little's formula on Eq. (5), we derive the average time to traverse packet in Model SPE given as,

$$t_{SPE} = \frac{E[L]_{SPE}}{T_{SPE}}, \tag{6}$$

where T_{SPE} is the total throughput of Model SPE and given as, $T_{SPE} = T_{cs} + (1-\beta)T_{es}$.

The average packet loss probability of Class CS (PL_{cs}) and Class ES (PL_{es}) represents the average number of packets being blocked or dropped by Class CS and Class ES out of total incoming packets. The probabilities PL_{cs} and PL_{es} for Model SPE are expressed as:

$$PL_{cs} = 1 - T_{cs}/T_c,$$
$$PL_{es} = 1 - T_{es}/\lambda_1. \tag{7}$$

The average packet loss probability for Model SPE is given as,

$$PL_{SPE} = 1 - T_{SPE}/\lambda_1. \tag{8}$$

The validity of the expressions given for SPE are contingent on the stability condition which can be derived using the traffic equilibrium equation (see pages 76–90 of (Gelenbe and Mitrani, 2010)). Let Tr_c, Tr_{cs} and Tr_{es} represent the traffic intensities at the controller, Class CS and Class ES of the switch respectively. The queueing network represented by Model SPE is stable if $(Tr_c - \mu_c) < 0$. Using the traffic equilibrium equations we get the following equality:

$$Tr_c = \beta Tr_{es}, \quad Tr_{cs} = Tr_c, \quad Tr_{es} = \lambda_1.$$

Solving the traffic equations for Model SPE and using the condition of $(Tr_c - \mu_c) < 0$, we get the stability condition for Model SPE as,

$$\beta\lambda_1 - \mu_c < 0. \tag{9}$$

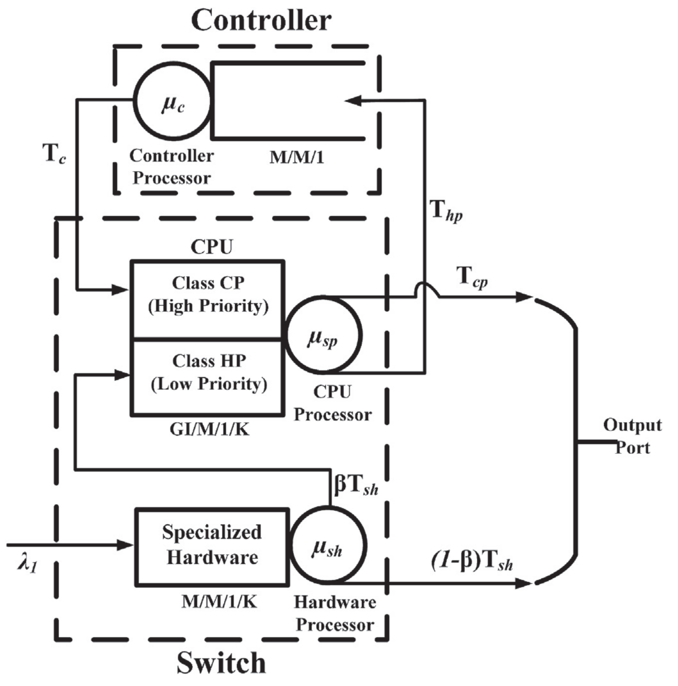


Fig. 4. Model HPE – hardware switch modelled with two servers to reflect the presence of network processing functions.

6. Model HPE: SDN hardware switch with priority queueing

Fig. 4 shows the model that uses priority queues for the CPU of hardware switch with finite capacity. The priority queues provides isolation between the packets arriving from the controller and packet to be processed by CPU when there is no matching FTE in hardware table stored in the switch TCAM. In this model, non-preemptive priority queues are used for CPU similar to that in Model SPE for software switch.

Class HP represents the low priority class of CPU for the external data packet that has no matching entry in the hardware table maintained by switch specialized hardware. This packet is sent to the controller by switch specialized hardware with the probability β. Class CP represents the high priority class for packets fed back to the CPU from the controller and must be forwarded out to the destination. It is assumed that CPU synchronizes the table information with specialized hardware as shown in Fig. 2. Both Class CP and Class HP queues shares service rate μ_{sp} while switch hardware queue has the service rate of μ_{sh}.

The HPE is modelled as a continuous time Markov chain $\{(n_c(t), n_{cp}(t), n_{hp}(t), n_{sh}(t)), t \geq 0\}$ on the state space $\{(w,x,y,z); w \in \mathbb{Z}_+, x \in \mathbb{Z}_+^{\leq K_1}, y \in \mathbb{Z}_+^{\leq K_2}, z \in \mathbb{Z}_+^{\leq K3}\}$, where $n_c(t)$, $n_{cp}(t)$, $n_{hp}(t)$ and $n_{sh}(t)$ are the number of packets in controller, class CP, class HP and switch hardware, respectively at time t. At time t, the random processes $\{n_c(t), n_{cp}(t), n_{hp}(t), n_{sh}(t)\}$ take on values $\{w,x,y,z\}$ and we have the following:

$$\{n_c(t), n_{cp}(t), n_{hp}(t), n_{sh}(t)\} = \{w,x,y,z\}. \tag{10}$$

The queue capacity for CPU of the switch is equal to K. It is assumed that Class CP and Class HP has the queue capacity of K_1 and K_2 respectively with the total queue capacity of CPU as K. Similarly, the queue capacity for switch hardware is equal to K_3.

The permissible transitions for the Markov chain $\{(n_c, n_{cp}, n_{hp}, n_{sh})\}$ are shown in Table 3 and these help us derive the sub-matrices (denoted by A_0, A_1, B_1 and A_2) of transition rate generator matrix determine (Q) for Model HPE which is used to compute the stationary distribution probability (π) for HPE.

In the following four subsections, we specify the expressions for the elements of sub-matrices A_0, A_1, B_1 and A_2 using standard QBD notation (Neuts, 1994).

Table 3
Permissible transitions for model HPE.

Event	From	To	Rate
One packet arrives at the switch hardware	$(n_c, n_{cp}, n_{hp}, n_{sh})$	$(n_c, n_{cp}, n_{hp}, n_{sh} + 1)$	λ_1
One packet departs from hardware to out of system	$(n_c, n_{cp}, n_{hp}, n_{sh})$	$(n_c, n_{cp}, n_{hp}, n_{sh} - 1)$	$\mu_{sh}(1 - \beta)$
One packet arrives at Class HP for CPU processing	$(n_c, n_{cp}, n_{hp}, n_{sh})$	$(n_c, n_{cp}, n_{hp} + 1, n_{sh} - 1)$	$\mu_{sh}\beta$
One packet forwarded from Class HP to controller	$(n_c, 0, n_{hp}, n_{sh})$	$(n_c, 0, n_{hp} - 1, n_{sh})$	μ_{sp}
One packet serviced by Controller to Class CP	$(n_c, n_{cp}, n_{hp}, n_{sh})$	$(n_c, n_{cp} + 1, n_{hp}, n_{sh})$	μ_c
One packet processed by CPU to out of system	$(n_c, n_{cp}, n_{hp}, n_{sh})$	$(n_c, n_{cp} - 1, n_{hp}, n_{sh})$	μ_{sp}

6.1. Elements of matrix A_0

The sub-matrix A_0 represents the phase distribution of Class CP, Class HP and switch hardware when the number of packets in controller (i.e. $n_c(t)$ or w in Eq. (10)) increases by 1.

$$A_{0(x,x')} = \begin{cases} A_{01}^{(x)}, & x' = x = 0, \\ 0, & \text{otherwise.} \end{cases}$$

where,

$$A_{01}{}^{(0)}_{(y,y')} = \begin{cases} A_{012}^{(y)}, & y' = y - 1, \\ 0, & \text{otherwise.} \end{cases}$$

where,

$$A_{012}{}^{(1 \le y \le K_2+1)}_{(z,z')} = \begin{cases} \mu_{sp}, & z' = z - 1, \\ 0, & \text{otherwise.} \end{cases}$$

6.2. Elements of matrix A_1

The sub-matrix A_1 represents the phase distribution of Class CP, Class HP and switch hardware when the number of packets in controller remain unchanged and there are some packets in the controller (i.e. $n_c(t)$ or w in Eq. (10) is a positive integer that remain unchanged).

$$A_{1(x,x')} = \begin{cases} A_{11}{}^{(x)}, & x' = x, \\ A_{12}{}^{(x)}, & x' = x - 1, \\ 0, & \text{otherwise.} \end{cases}$$

where,

$$A_{11}{}^{(0 \le x \le K_1+1)}_{(y,y')} = \begin{cases} A_{110}^{(y)}, & y' = y + 1, \\ A_{111}^{(y)}, & y' = y, x = 0, \\ \widetilde{A}_{111}^{(y)}, & y' = y, x \ne 0, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$A_{12}{}^{(1 \le x \le K_1+1)}_{(y,y')} = \begin{cases} A_{121}^{(y)}, & y' = y, \\ 0, & \text{otherwise.} \end{cases}$$

where,

$$A_{111}{}^{(0 \le y \le K_2+1)}_{(z,z')} = \begin{cases} \lambda_1, & z' = z + 1, \\ \mu_{sh}(1 - \beta), & y = 0, z' = z - 1, \\ 0, & \text{otherwise,} \end{cases}$$

$$\widetilde{A}_{111}^{(0 \le y \le K_2+1)} = A_{111}^{(0 \le y \le K_2+1)},$$

$$A_{110}{}^{(0 \le y < K_2+1)}_{(z,z')} = \begin{cases} \mu_{sh}\beta, & z' = z, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$A_{121}{}^{(0 \le y \le K_2+1)}_{(z,z')} = \begin{cases} \mu_{sp}, & z' = z, \\ 0, & \text{otherwise.} \end{cases}$$

The diagonal elements of A_{111} and $\widetilde{A}_{111}$ are given as,

$$A_{111}{}^{(y)}_{(z,z)} = \begin{cases} -\lambda_1 - \mu_c, & y = 0, z = 0, \\ -\lambda_1 - \mu_c - \mu_{sp}, & 0 < y \le K_2 + 1, z = 0, \\ -\lambda_1 - \mu_c - \mu_{sh}, & y = 0, 0 < z \le K_3, \\ -\lambda_1 - \mu_c - \mu_{sh} - \mu_{sp}, & 0 < y \le K_2 + 1 \text{ and,} \\ & 0 < z \le K_3 \\ -\mu_c - \mu_{sh}, & y = 0, z = K_3 + 1, \\ -\mu_c - \mu_{sh} - \mu_{sp}, & 0 < y \le K_2 + 1 \text{ and,} \\ & z = K_3 + 1, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$\widetilde{A_{111}}{}^{(y)}_{(z,z)} = \begin{cases} A_{111}^{(y)} - \mu_{sp} I, & y = 0, \\ A_{111}^{(y)}, & y \ne 0, \\ 0, & \text{otherwise.} \end{cases}$$

6.3. Elements of matrix A_2

The sub-matrix A_2 represents the phase distribution of Class CP, Class HP and switch hardware when the number of packets in controller (i.e. $n_c(t)$ or w in Eq. (10)) decreases by 1.

$$A_{2(x,x')} = \begin{cases} A_{20}{}^{(x)}, & x' = x + 1, \\ 0, & \text{otherwise.} \end{cases}$$

where

$$A_{20}{}^{(0 \le x \le K_1+1)}_{(y,y')} = \begin{cases} A_{201}{}^{(y)}, & y' = y, \\ 0, & \text{otherwise.} \end{cases}$$

where,

$$A_{201}{}^{(0 \le y \le K_2+1)}_{(z,z')} = \begin{cases} \mu_c, & z' = z, \\ 0, & \text{otherwise.} \end{cases}$$

6.4. Elements of matrix B_1

The sub-matrix B_1 is identical to A_1 that represents the phase distributions of Class CP, Class HP and switch hardware when the number of packets in controller (i.e. $n_c(t)$ or w in Eq. (10) is equal to 0).
$\therefore B_1 = A_1(\text{for } \mu_c = 0)$.

6.5. Performance metrics

With these sub-block matrices and matrix geometric solution, we can compute the stationary distribution probabilities ($\pi_{i,j,k,l}$) for i packets in controller, j packets in Class CP, k packets in Class HP, and l packets in the switch hardware. The performance metrics like throughput of the controller and switch, the average time to install FTE in the switch, average packet loss in the switch can be computed using ($\pi_{i,j,k,l}$).

The throughput of controller (T_c) for Model HPE is given by the sum of probabilities that the controller has at least one packet to forward to

Class CP with service rate of μ_c and this is given by:

$$T_c = \mu_c \sum_{i=1}^{\infty} \sum_{j=0}^{K_1} \sum_{k=0}^{K_2} \sum_{l=0}^{K_3} \pi_{i,j,k,l}. \tag{11}$$

Similarly, the throughput of Class CP (T_{cp}) is given by the sum of probabilities that the Class CP has at least one packet to process with service rate of μ_{sp} and this is given by:

$$T_{cp} = \mu_{sp} \sum_{i=0}^{\infty} \sum_{j=1}^{K_1} \sum_{k=0}^{K_2} \sum_{l=0}^{K_3} \pi_{i,j,k,l}. \tag{12}$$

Similarly, the throughput of Class HP (T_{hp}) is given by the sum of probabilities that the Class HP has at least one packet to forward to the controller with service rate of μ_{sp} and no packet in Class CP in the stationary state, and this is given by:

$$T_{hp} = \mu_{sp} \sum_{i=0}^{\infty} \sum_{k=1}^{K_2} \sum_{l=0}^{K_3} \pi_{i,0,k,l}. \tag{13}$$

Also, the throughput of the switch hardware (T_{sh}) is given by the sum of probabilities that switch hardware has at least one packet to forward with service rate of μ_{sh} and this is given by:

$$T_{sh} = \mu_{sh} \sum_{i=0}^{\infty} \sum_{j=0}^{K_1} \sum_{k=0}^{K_2} \sum_{l=1}^{K_3} \pi_{i,j,k,l}. \tag{14}$$

The average number of total packets for Model HPE is given as,

$$E[L]_{HPE} = \sum_{i=0}^{\infty} \sum_{j=0}^{K_1} \sum_{k=0}^{K_2} \sum_{l=0}^{K_3} (i+j+k+l)\pi_{i,j,k,l}. \tag{15}$$

Applying Little's formula on Eq. (15), we derive the average time to traverse packet in Model HPE given as,

$$t_{HPE} = \frac{E[L]_{HPE}}{T_{HPE}}, \tag{16}$$

where T_{HPE} is the total throughput of Model HPE and given as, $T_{HPE} = T_{cp} + (1 - \beta)T_{sh}$.

The average packet loss probability of Class CP (PL_{cp}), Class HP (PL_{hp}) and switch hardware (PL_{sh}) represents the average number of packets being blocked or dropped by Class CP, Class HP and switch hardware out of total incoming packets in respective queue. PL_{cp}, PL_{hp} and PL_{sh} for Model HPE are expressed as,

$$PL_{cp} = 1 - T_{cp}/T_c,$$

$$PL_{hp} = 1 - T_{hp}/T_{sh},$$

$$PL_{sh} = 1 - T_{sh}/\lambda_1. \tag{17}$$

The average throughput and packet loss probability for Model HPE are given as,

$$T_{HPE} = T_{cp} + (1-\beta)T_{sh}, \tag{18}$$

$$PL_{HPE} = 1 - T_{HPE}/\lambda_1. \tag{19}$$

The validity of the expressions given for HPE are contingent on the stability condition which can be derived using the traffic equilibrium equation (Gelenbe and Mitrani, 2010). Let Tr_c, Tr_{cp}, Tr_{hp} and Tr_{sh} denote the traffic intensities at the controller, Class CP, Class ES of the CPU and switch hardware respectively. The queueing network represented by Model HPE is stable if $(Tr_c - \mu_c) < 0$. Using the traffic equilibrium equations, we get,

$$Tr_c = Tr_{hp}, \quad Tr_{cp} = Tr_c, \quad Tr_{hp} = \beta Tr_{sh}, \quad Tr_{sh} = \lambda_1.$$

Solving the traffic equations for Model HPE and using the condition of $(Tr_c - \mu_c) < 0$, we get the stability condition for Model HPE which is same as that of Model SPE, i.e.,

$$\beta\lambda_1 - \mu_c < 0. \tag{20}$$

Table 4
Parameter used for Model SPE vs. HPE.

Parameter	Value
Table miss probability, β	$0.1 \sim 1$
Switch CPU processing rate, μ_{sp} (packets/sec)	1000
Switch hardware processing rate, μ_{sh} (packets/sec)	$1000 \times \mu_{sp}$
Controller to CPU Processing Ratio, m_r	$0.1 \sim 2$
Switch Hardware to CPU Processing Ratio, m_s	$1 \sim 1000$
External Arrival rate, λ_1 (packets/sec)	$120 \sim 480$
Queue capacity for Class ES or HP, K_2	10
Queue capacity for Class CS or CP, K_1	50% of K_2
Queue capacity for Switch hardware, K_3	20% of K_2

7. Results

In order to make a fair comparison between SPE and HPE, we use identical parameters for analysis and simulation of both Model SPE for Model HPE. The parameters used for the analysis and simulation results are shown in Table 4. The analytical and simulation results in this paper are obtained for parameters β, λ_1 and μ_c that satisfy the stability conditions for both SPE and HPE (Eq. (9) and Eq. (20)).

The CPU processor (μ_{sp}) of Model SPE is assumed as 1000 packets/sec. For Model HPE, the switch hardware is typically much faster than CPU processor. We introduce the parameter m_s to denote the ratio between the service rate of the switch hardware to the service rate of the CPU (both in packets per second). The parameter m_s is varied from 1 to 1000 to test the impact of processing rate disparity in HPE. The table miss probability (β) and the controller to CPU processing ratio (m_r) are assumed to be in the range of $0.1 \sim 1$ and $0.1 \sim 2$ respectively.

External packets arrive to Class ES of Model SPE, and the queue capacity of Class ES (K_2) is assumed as 10. The packets fed back by the controller to Class CS is the fraction of packets arriving at Class ES. Hence, it is assumed that queue capacity of Class CS (K_2) is 50% of Class ES.

For Model HPE, the value of K_1 is used for Class HP and K_2 is used for Class CP. It is assumed that the queue capacity of switch hardware (K_3) is a fraction of K_2, i.e. K_3 is 20% of K_2 because the CPU processor is able to handle larger number of flow requests than hardware processor (though not speedier). In this paper, the external arrival rate at Class ES of Model SPE and switch hardware of Model HPE is assumed to be in the range of ($120 \sim 480$) packets/sec reflecting typical arrival rates in campus networks (Zhao and Hua, 2014) and residential area gateways (Heegaard, 2007).

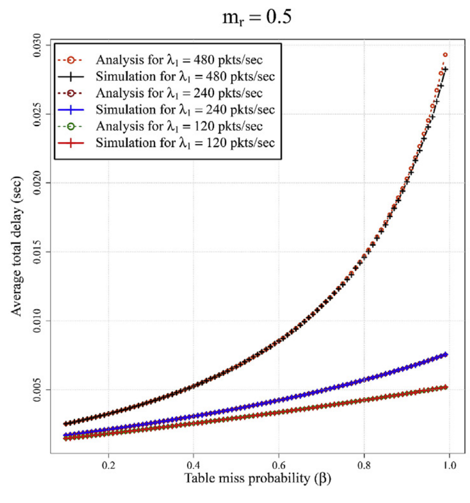
The simulations are repeated hundred times and the 95% confidence intervals (CI) are computed on the basis that the errors are normally distributed. In the following section, we validate the accuracy of analytical results for both models SPE and HPE, followed by relative total average packet transfer delay and packet loss probability between models SPE and HPE, and design guidelines based on the analytical results.

7.1. Accuracy: analytical vs. simulation

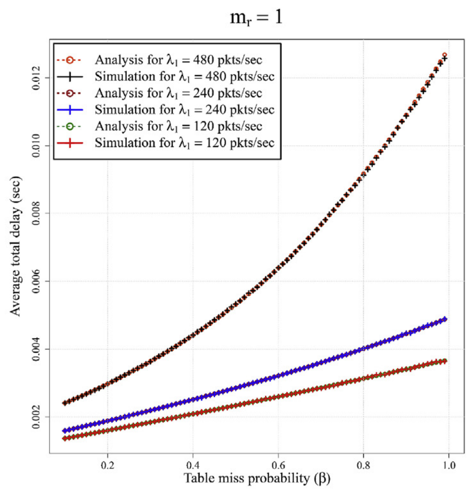
In this section, we validate the accuracy of analytical results for models SPE and HPE by comparing them with simulation results. The performance metrics used for validating the accuracy of analytical results are average total time (or delay) for the packets to traverse out of the system, and total packet loss probability.

The analytical results for the average time packets are held in the switch for Model SPE (denoted by t_{SPE} as in Eq. (6)) and Model HPE (denoted by t_{HPE} as in Eq. (16)) is computed respectively. Similarly, the analytical results for the packet loss probability in Model SPE (denoted by PL_{SPE} as in Eq. (8)) and HPE (denoted by PL_{HPE} as in Eq. (19)) is computed respectively.

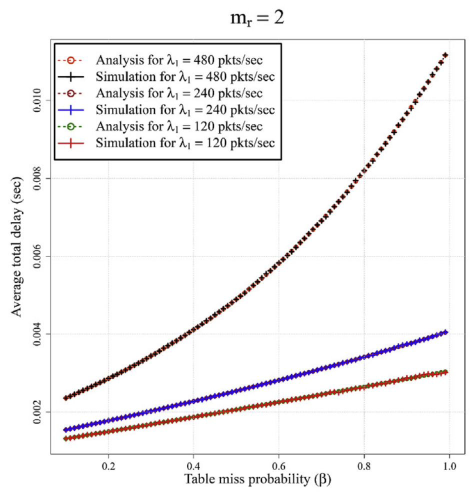
Fig. 5a–c show the predicted average delay while Fig. 5d–f show the predicted packet loss of model SPE compared to discrete event simulations. In each sub-figure the predictions from the analytical model



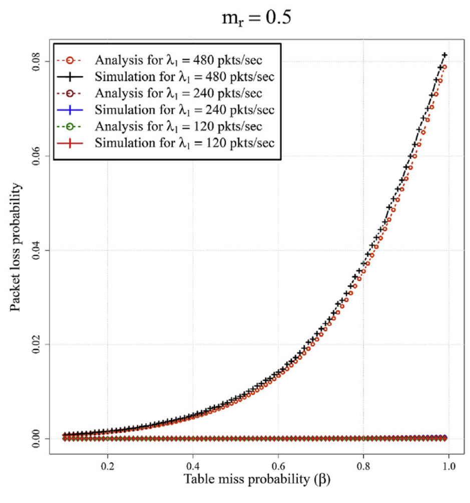
(a)



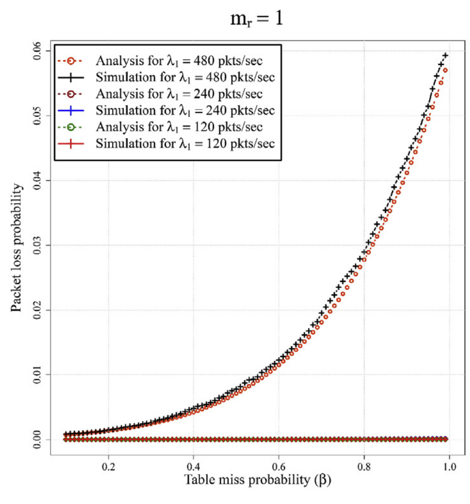
(b)



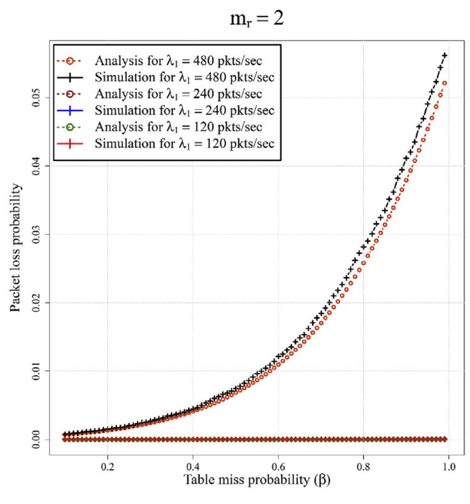
(c)



(d)



(e)



(f)

Fig. 5. Accuracy of Model SPE analytical results for average total delay to transfer packets (a–c) and packet loss probability (d–e) for μ_{sp} =1000 pkts/sec and increasing β.

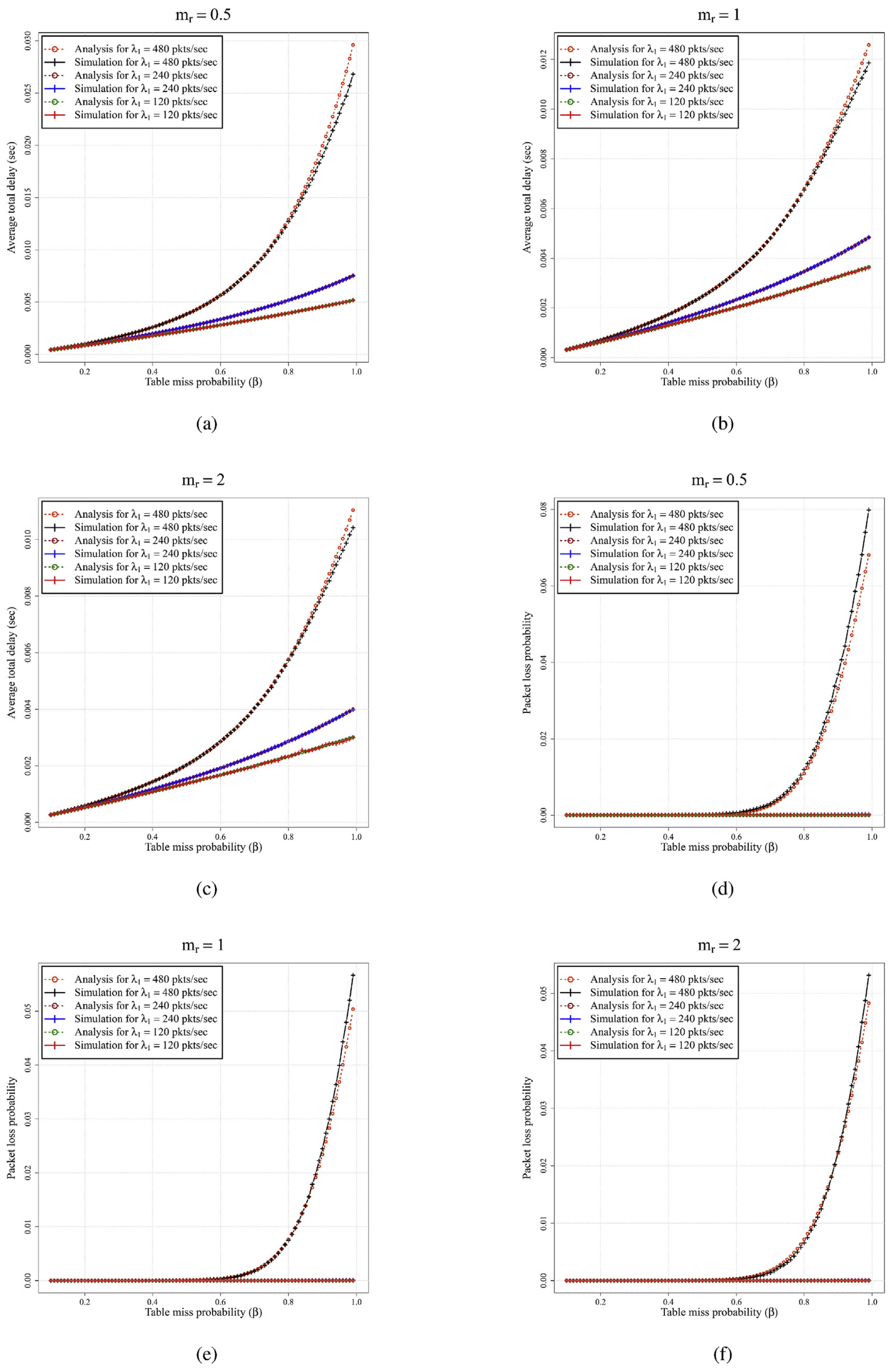


Fig. 6. Accuracy of Model HPE analytical results for average total delay to transfer packets (a–c) and packet loss probability (d–e) for μ_{sp} = 1000 pkts/sec and increasing β.

for SPE tracks the simulations well with increasing β. Inspecting across sub-figures, the model predictions are in good agreement with simulations for increasing m_r. The highest error recorded by SPE (compared to simulation) is 4% (in terms of packet loss probability) and 0.2% (in terms of average total delay), and this occurs in Fig. 5d.

For model HPE a similar observation is made based on the comparisons between the analytical predictions of HPE against discrete event simulation results in Fig. 6a–f. Larger errors were recorded with higher values of β and m_r, potentially due to interactions between the controller and the switch that are present in the discrete event simulations but not in the analytical models. Overall, the analytical model for HPE tracks the trend and values of the simulations correctly. The maximum difference observed for model HPE compared with simulation is 4.6% (in terms of packet loss probability) and 0.3% (in terms of average total delay).

Both Figs. 5 and 6 shows the accuracy of analytical results of Model SPE and Model HPE respectively. The simulation results were very close to the analytical results for both total average delay and packet loss probability to validate the accuracy of the analytical models. The agreement in trend for the analytical results and also prediction errors below 5% indicate that the model is correct and accurately predicts switch performance compared to simulations.

The sub-sections that follow will only show the relative analytical comparisons between models SPE and HPE to provide better understanding and guidelines to the network engineer on the merits of hardware vs. software switches. Additionally, we show results for $\lambda_1 = 480$ from this point forward for clarity and ease of presentation. Results for $\lambda_1 = \{120, 240, 720, 960\}$ show similar trends and characteristics and thus offer no additional insights to what is shown with $\lambda_1 = 480$.

7.2. Relative average total packet transfer delay

In Fig. 7 we compare the average time packets are in the switch between SPE (denoted by t_{SPE} as in Eq. (6)) and HPE (denoted by t_{HPE} as in Eq. (16)). In this comparison, we are investigating the effects of delay in software and hardware switch.

The relative time difference (denoted by ϵ_t) to traverse packets in the switch between SPE and HPE (both with finite capacity) is calculated as:

$$\epsilon_t = \frac{(t_{SPE} - t_{HPE})}{t_{SPE}} \times 100\%.$$

Fig. 7a and b shows the average relative time for packets to traverse a switch between SPE and HPE for varying β, m_r and m_s respectively. From Fig. 7a, the relative average packet delay decreases 85%–1.8% for increasing β. However, the trend is reversed in Fig. 7b with increasing m_r, whereby the relative total delay increases from 0.8% to 56% as m_r increases from 0.5 to 2.0.

As the value β increases, the traffic processed by CPU in Model HPE significantly increases, therefore increasing the delay predicted by HPE and diminishing the benefit of having dedicated switch hardware for forwarding.

Moreover, as m_r increases, the relative average delay between HPE and SPE increases exponentially and then plateaus off. The reason for this is that higher m_r means controller processing power is higher than CPU processor at the switch which significantly reduces the delay caused by packets traversing the control path.

7.3. Relative average packet loss probability

In this section, we compare the packet loss probability between SPE (denoted by PL_{SPE} as in Eq. (8)) and HPE (denoted by PL_{HPE} as in Eq. (19)). In this comparison, we are investigating the effects of packet loss probability in software and hardware switch.

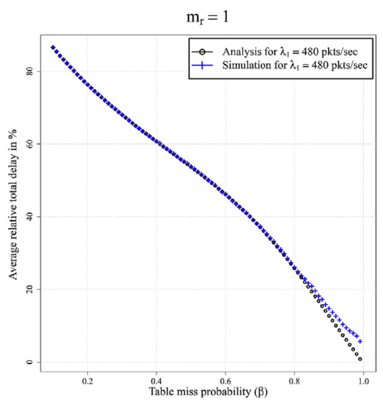
The relative packet loss probability difference (denoted by ϵ_{PL}) in the switch between SPE and HPE is calculated as:

$$\epsilon_{PL} = \frac{(PL_{SPE} - PL_{HPE})}{PL_{SPE}} \times 100\%.$$

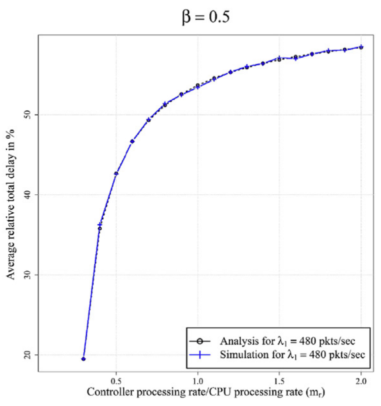
Fig. 8 shows the average packet loss probability between models SPE and HPE for increasing values of β, m_r and m_s respectively. From Fig. 8a, the relative packet loss probability is up to 100% higher in Model SPE than Model HPE for lower β, and it decreases to approximately 2% for $\beta = 1.0$.

The reason for this is, as the β increases, the amount of traffic processed by the CPU in Model HPE significantly increases (i.e. the hardware processing at the switch is not leveraged) rendering its performance closer to that of SPE, therefore reducing the gap between models SPE and HPE for packet loss probability. This difference in performance decreases drastically for higher β, for which the packet loss probability is much higher due to increasing amount of traffic to be processed by the CPU.

The trend for the relative packet loss probabilities in Fig. 8a and b are quite similar to the relative total delays in Fig. 7a and b. The relative packet loss probability is up to 83% higher in Model SPE than Model HPE for increasing m_r and remains a steady 100% with increasing m_s. The reason for increasing relative packet loss probability with increasing m_r is: a higher m_r means higher controller processing power than CPU processor which quickly feedback the packet to CPU, significantly increasing the packet loss probability in CPU.



(a)



(b)

Fig. 7. Average relative total delay to transfer packets between Model SPE and Model HPE for $\lambda_1 = 480$ pkts/sec and $\mu_{sp} = 1000$ pkts/sec for (a) increasing β; (b) and increasing m_r.

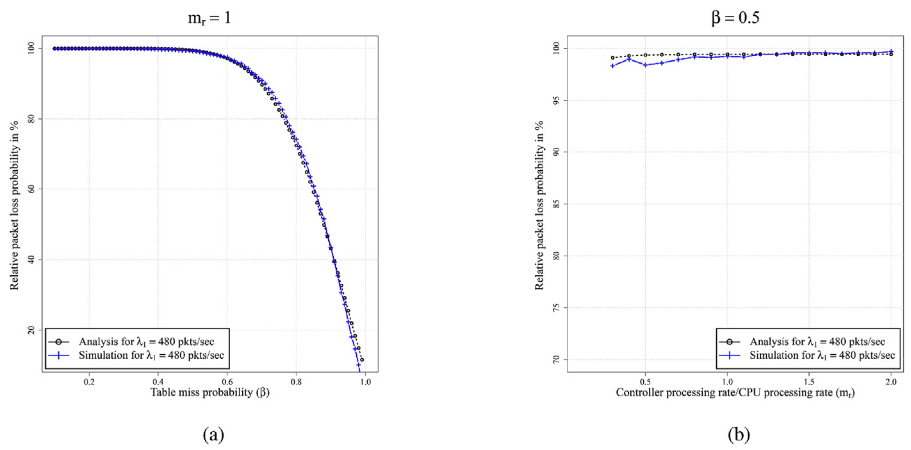


Fig. 8. Relative packet loss probability between Model SPE and Model HPE for $\lambda_1 = 480$ pkts/sec and $\mu_{sp} = 1000$ pkts/sec for (a) increasing β; (b) and increasing m_r.

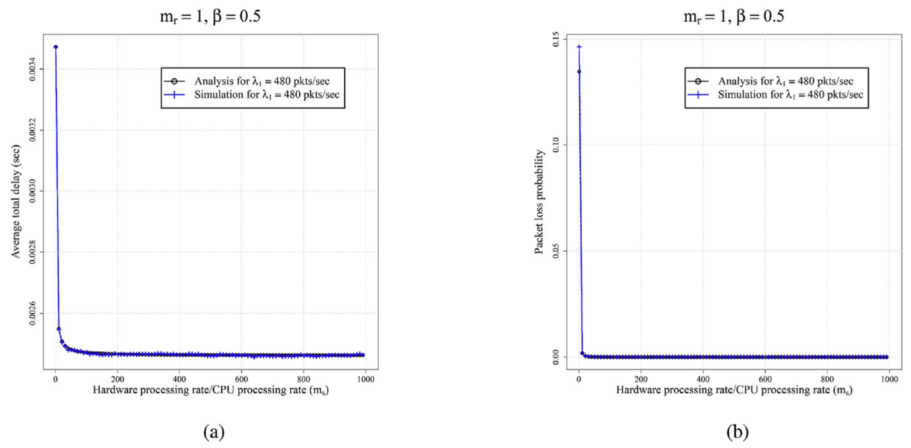


Fig. 9. Effect of varying switch hardware processor in Model HPE for $\lambda_1 = 480$ pkts/sec and $\mu_{sp} = 1000$ pkts/sec for increasing m_s.

7.4. *Effect of varying μ_{sh} in model HPE*

In this section, we study the effect of varying the ratio of packet forwarding rate of the switch hardware to the CPU (m_s) for Model HPE. Recall from earlier discussion in Section 7.2, m_s is only defined for HPE. Fig. 9a and b show the average delay for the packet to traverse and packet loss probability for increasing m_s, respectively. The results show that both the average delay and packet loss probability for Model HPE is steady for m_s greater than 100 and appears to be insensitive to increasing m_s. Over provisioning the switch hardware does not confer any benefits to improving QoS.

7.5. *Operational guidelines*

Based on the results in Figs. 7 and 8, it is clear that Model HPE provides lower average total packet delay and lower packet loss probability compared to Model SPE. As β increases more packets recourse to the controller and traverse the control path thus narrowing the performance difference between HPE and SPE.

Considering these factors, we provide the following points for consideration between Model SPE (software switch) or Model HPE (hardware switch) in SDN:

(i) For delay sensitive applications, hardware switches significantly reduces the total delay by at least 80% when the number of new flows entering the switch is below 40%.
(ii) For loss sensitive applications, a hardware switch achieves up to 100% lower packet loss probabilities compared to a software switch.

8. Conclusion

In this study, we compared a software switch (SPE) and hardware switch model (HPE) in SDN using an analytical approach to provide insights on the performance of software and hardware switches. From our investigations, we find that a hardware switch performs better than a software switch in terms of average delay and packet loss probability. This is contingent on a stable network whereby the number of new flows that arrive at a switch for decisioning is low. Increasing involvement of the controller means that the hardware switch increasingly uses the CPU for forwarding (as opposed to ASICs and TCAM) hence reducing the benefits of using a hardware switch. Our future work involves validation of these models experimentally with the measurement that can provide realistic detail insights.

Acknowledgement

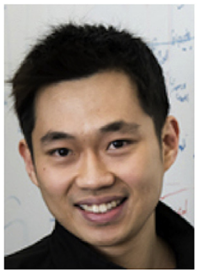
This research is partly supported by Victoria's Huawei NZ Research Programme, Software-Defined Green Internet of Things project (E2881) and Victoria Doctoral Scholarship.

References

Azodolmolky, S., Nejabati, R., Pazouki, M., Wieder, P., Yahyapour, R., Simeonidou, D., 2013. An analytical model for software defined networking: a network calculus-based approach. In: Global Communications Conference (GLOBECOM). IEEE, pp. 1397–1402.
Burke, P.J., 1956. The output of a queuing system. Oper. Res. 4 (6), 699–704.

Busic, A., Gaujal, B., Perronnin, F., 2012. Perfect sampling of networks with finite and infinite capacity queues. In: Al-Begain, K., Fiems, D., Vincent, J.-M. (Eds.), ASMTA, Vol. 7314 of Lecture Notes in Computer Science. Springer, pp. 136–149.
Gelenbe, E., Mitrani, I., 2010. Analysis and Synthesis of Computer Systems, vol. 4. World Scientific.
Goransson, P., Black, C., 2014. Software Defined Networks: a Comprehensive Approach. Elsevier.
Goto, Y., Masuyama, H., Ng, B., Seah, W.K., Takahashi, Y., 2016. Queueing analysis of software defined network with realistic openflow–based switch model. In: Proceedings of the IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), London, UK.
Haiyan, M., Jinyao, Y., Georgopoulos, P., Plattner, B., 2016. Towards SDN based queuing delay estimation. China Commun. 13 (3), 27–36.
Heegaard, P.E., 2007. Evolution of traffic patterns in telecommunication systems. In: Communications and Networking in China, 2007. CHINACOM'07. Second International Conference on. IEEE, pp. 28–32.
Huang, D.Y., Yocum, K., Snoeren, A.C., 2013. High-fidelity switch models for software-defined network emulation. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 43–48.
Huang, J., Xu, L., Duan, Q., Xing, C.-c., Luo, J., Yu, S., 2017. Modeling and performance analysis for multimedia data flows scheduling in software defined networks. J. Netw. Comput. Appl. 83, 89–100.
Indigo: Open Source Openflow Switches. https://floodlight.atlassian.net/wiki/spaces/HOME/overview.
Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P., 2011. Modeling and performance evaluation of an OpenFlow architecture. In: Proceedings of the 23rd International Teletraffic Congress. International Teletraffic Congress, pp. 1–7.
Javed, U., Iqbal, A., Saleh, S., Haider, S.A., Ilyas, M.U., 2017. A stochastic model for transit latency in OpenFlow SDNs. Comput. Network. 113, 218–229.
Koohanestani, A.K., Osgouei, A.G., Saidi, H., Fanian, A., 2017. An analytical model for delay bound of OpenFlow based SDN using network calculus. J. Netw. Comput. Appl. 96, 31–38.
Kumar, R., Hasan, M., Padhy, S., Evchenko, K., Piramanayagam, L., Mohan, S., Bobba, R.B., 2017. End-to-End network delay guarantees for real-time systems using SDN. In: IEEE Conference Real-time Systems Symposium (RTSS) (Accepted).
Kuźniar, M., Perešíni, P., Kostić, D., 2015. What you need to know about sdn flow tables. In: International Conference on Passive and Active Network Measurement. Springer, pp. 347–359.
Mahmood, K., Chilwan, A., Østerbø, O.N., Jarschel, M., 2014. On the modeling of openflow-based SDNs: the single node case. In: Proceedings of the 6th International Conference on Networks and Communications (NeCoM 2014), pp. 207–214 arXiv:1411.4733v1.
Miao, W., Min, G., Wu, Y., Wang, H., 2015. Performance modelling of preemption-based packet scheduling for data plane in software defined networks. In: IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 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 12 (5s), 77.
Mondal, A., Misra, S., Maity, I., 2018. Buffer size evaluation of OpenFlow systems in software-defined networks. IEEE Syst. J. 1–8.
Nayak, N.G., Dürr, F., Rothermel, K., 2016. Time-sensitive software-defined network (TSSDN) for real-time applications. In: Proceedings of the 24th International Conference on Real-time Networks and Systems. ACM, pp. 193–202.
Neuts, M.F., 1994. Matrix-geometric Solutions in Stochastic Models - an Algorithmic Approach. Dover Publications.
Nunes, B., Mendonca, M., Xuan-Nam Nguyen, O., K, T., 2014. A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surveys Tutor. 16 (3), 1617–1634.
ofsoftswitch13. https://github.com/CPqD/ofsoftswitch13.
ONF, 2014. Open Networking Foundation.
Open vSwitch. http://openvswitch.org/.
Pan, H., Guan, H., Liu, J., Ding, W., Lin, C., Xie, G., 2013. The FlowAdapter: enable flexible multi-table processing on legacy hardware. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 85–90.
Pantou: OpenFlow 1.3 for OpenWRT. https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT.
Rawat, D.B., Reddy, S.R., 2017. Software defined networking architecture, security and energy efficiency: a survey. IEEE Commun. Surveys Tutor. 19 (1), 325–346, https://doi.org/10.1109/COMST.2016.2618874.
P. Rygielski, M. Seliuchenko, S. Kounev, M. Klymash, Performance Analysis of SDN Switches with Hardware and Software Flow Tables.
Z. Shang, K. Wolter, Delay Evaluation of OpenFlow Network Based on Queueing Model, arXiv preprint arXiv:1608.06491.
Singh, D., Ng, B., Lai, Y.-C., Lin, Y.-D., Seah, W.K.G., 2017. Modelling software-defined networking: switch design with finite buffer and priority queueing. In: 42nd Annual IEEE Conference on Local Computer Networks (LCN 2017), Singapore, Singapore.
Sood, K., Yu, S., Xiang, Y., 2016. Performance analysis of software-defined network switch using *M/Geo/*1 model. IEEE Commun. Lett. 20 (12), 2522–2525.
Takagi, H., 1990. Stochastic Analysis of Computer and Communication Systems. Elsevier Science Inc.
Xiong, B., Yang, K., Zhao, J., Li, W., Li, K., 2016. Performance evaluation of OpenFlow-based software-defined networks based on queueing model. Comput. Network. 102, 172–185.
Yen, T.-C., Su, C.-S., 2014. An SDN-based cloud computing architecture and its mathematical model. In: Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on, vol. 3. IEEE, pp. 1728–1731.
Zhao, C., Hua, C., 2014. Traffic-load aware user association in dense unsaturated wireless networks. In: 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6, https://doi.org/10.1109/WCSP.2014.6992149.

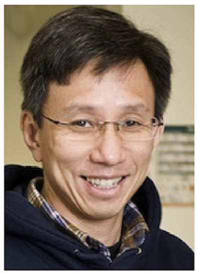
Deepak Kumar Singh received the B.Eng. degree in Electronics and Communication engineering from Kathmandu Engineering College (Affiliated to Tribhuvan University), Kathmandu, Nepal in 2010, the M.Eng. degree in Electronics and Radio engineering from Kyung Hee University, Gyeonggi-do, South Korea, in 2014. He is currently working towards the Ph.D. degree at Victoria University of Wellington, New Zealand. His research focuses on modelling of Software-Defined Network.



Bryan Ng completed his PhD (2010) in the area of communication and networking. He held teaching & research positions in Malaysia and France in addition to attachments to commercial research laboratories Intel, Motorola, Panasonic and Orange Labs. His research interest include performance analysis of communication networks, modelling networking protocols and software defined networking.

Yuan-Cheng Lai received his Ph.D. degree in the Department of Computer and Information 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 August 2001 and has been a professor since February 2008. His research interests include performance analysis, protocol design, wireless networks, and and network security.

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. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), and ONF Research Associate, and is the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST). He published a textbook, Computer Networks: An Open Source Approach (McGraw-Hill, 2011).



Winston K.G. Seah received the Dr.Eng. degree from Kyoto University, Kyoto, Japan, in 1997. He is currently Professor of Network Engineering in the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Prior to this, he has worked for more than 16 years in mission-oriented industrial research, taking ideas from theory to prototypes, most recently, as a Senior Scientist in the Institute for Infocomm Research, Singapore. His latest research interests include Internet of Things, wireless sensor networks powered by ambient energy harvesting, wireless multi-hop networks, software defined networking, and 5G access protocols for machine-type communications.