Future Generation Computer Systems 106 (2020) 412–425

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Modeling and optimization of packet forwarding performance in software-defined WAN

Jinyuan Zhao [a,b], Zhigang Hu [a,*], Bing Xiong [c], Liu Yang [a], Keqin Li [d]

[a] School of Computer Science and Engineering, Central South University, Changsha 410075, PR China
[b] School of Computer and Communication, Hunan Institute of Engineering, Xiangtan 411104, PR China
[c] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, PR China
[d] Department of Computer Science, State University of New York at New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 21 June 2019
Received in revised form 21 August 2019
Accepted 6 December 2019
Available online 7 January 2020

MSC:
00-01
99-00

Keywords:
Controller cluster deployments
OpenFlow switches
Optimization models
Queueing system
Software-defined networking
Wide-area networks

ABSTRACT

As a novel network paradigm, Software-Defined Networking (SDN) offers numerous benefits for wide-area networks (WAN), like promoting application performance and reducing deployment costs. However, it also comes along with an inherent penalty to essential network performance such as packet forwarding delay, primarily due to the involvement of logically centralized controllers. This paper is motivated to provide an accurate queueing system of packet forwarding performance in software-defined WAN based on modeling its controller cluster and OpenFlow switches. In particular, we approximate the packet-in message processing of the controller cluster as an $M/M/n$ queue based on the derivation of its message arrival process. Meanwhile, we characterize the packet processing of an OpenFlow switch as an $M/G/1$ queue after taking an insight into its packet switching process. As a further step, we build an optimization model of controller cluster deployments to obtain the optimal number of controllers in the cluster. Finally, our proposed queueing model of SDN controller cluster is evaluated with the prevalent benchmark OFsuite_Performance by experiments, and their results indicate that our proposed model provides a more accurate approximation of controller cluster performance. Furthermore, we perform numerical analysis on packet forwarding delay and solve the optimal number of controllers for different varying parameters, which offer effective guidelines for software-defined WAN deployments.

1. Introduction

As a novel network paradigm, Software-Defined Networking (SDN) separates network control from data forwarding devices, and allows for a separate controller entity to manipulate substrate switches through southbound interface typically OpenFlow [1,2]. The paradigm paves the way for a more flexible, programmable, and innovative networking, and is commonly regarded as one of the promising directions towards future Internet. Over the past several years, OpenFlow-based SDN concept has already come into implementations in a variety of production networks such as wide-area networks (WAN), and provided numerous benefits like enhancing data transfer efficiency, promoting application performance, and reducing deployment costs [3,4]. As a pioneer enterprise, Google designed and implemented a private software-defined WAN (SD-WAN) called B4 connecting its data centers across the planet as early as 2013 [5], and incrementally moved from offering best-effort content-copy services to carrier-grade availability for the following 5-year evolution [6].

In SDN architecture, the control plane is responsible for constructing a global network view based on topology discovery, and administrating data forwarding behaviors of all underlying switches. This is prone to incur serious performance bottlenecks of logically centralized control plane, and brings about an inherent penalty to essential network performance typically packet forwarding delay. For example, the NOX controller roughly handles 30 k flow setup requests per second at most in case of keeping flow setup time at 15 ms [7], while a data center with 100 switches can generate $1M$ flow setup requests per second at the worst [8]. This implies that a single controller is incapable of handling all flow setup requests from OpenFlow switches especially in SD-WAN scenarios. Thus it is imperative to deploy multiple controllers in the control plane typically in the form of a cluster. When it comes to SD-WAN deployments, it is a prerequisite to understand their network performance and limitation.

Queueing theory has been widely applied in network performance evaluation, due to its unique advantage of a quick performance approximation [9]. Initially, some researchers built queueing models of SDN controllers by characterizing their flow setup requests as batch arrival process [10,11]. However, the batch arrivals could not exactly describe the pattern of flow setup

* Corresponding author.
E-mail address: zghu@csu.edu.cn (Z. Hu).

https://doi.org/10.1016/j.future.2019.12.010
0167-739X/

requests from multiple switches. A few literatures concentrated on queueing models of OpenFlow switches with an arbitrary assumption that packet switching time conforms to exponential distribution [12,13] without a careful investigation into packet switching process. More close to our work, several analytical models of OpenFlow-based SDN approximate the data plane as an open Jackson network with the controller also modeled as a $M/M/1$ queue [14,15]. Nevertheless, these work was carried out on hypothetical queueing parameters lacking of deep insights into practical queueing characteristics in specific network scenarios.

For the above situations, this paper is motivated to provide an accurate queueing system of packet forwarding performance in software-defined WAN. To achieve this aim, we firstly depict a typical deployment scenario of software-defined WAN. Then, we model the packet processing of each switch and the packet-in message processing of the controller cluster, based on investigates into their queueing parameters. Subsequently, we construct a queueing system of packet forwarding to derive average packet delay through an OpenFlow switch. Furthermore, we build an optimization model of controller cluster deployments to solve the optimal number of controllers. Finally, we evaluate the queueing model of SDN controller clusters with the prevalent benchmark OFsuite_Performance, carry out numerical analysis on packet forwarding delay, and solve the optimal number of controllers for different varying parameters.

With the above methodology, this paper aims to achieve the following conclusions as our main contributions: (a) pointing out that the packet processing time of an OpenFlow switch conforms to general distribution due to its multiple processing steps; (b) concluding that packet-in message arrivals at SDN controller clusters can be characterized as Poisson stream in software-defined WAN; (c) building a queueing system of packet forwarding performance in software-defined WAN with its controller cluster and OpenFlow switches respectively modeled as $M/M/n$ and $M/G/1$ queues; (d) proposing an optimization model of controller cluster deployments, based on average packet forwarding delay through an OpenFlow switch; (e) solving the optimization model to obtain the optimal number of controllers in the cluster, based on the proof of the convexity of its objective function.

The reminder of the paper is organized as follows. Section 2 introduces related work. In Section 3, we give a typical deployment scenario of software-defined WAN. Section 4 models the packet-in message processing of a controller cluster as an $M/M/n$ queue based on the derivation of its message arrival process. In Section 5, we apply the $M/G/1$ model to characterize the packet processing of an OpenFlow switch based on investigations into its packet switching process. Section 6 formulates a queueing system of packet forwarding performance in software-defined WAN, and builds an optimization model of controller cluster deployments. In Section 7, we evaluate the queueing model of SDN controller clusters with the prevalent benchmark OFsuite_Performance, perform numerical analysis on packet forwarding delay, and solve the optimal number of controllers in terms of different parameters. Section 8 concludes the paper.

2. Related work

The SDN paradigm offers flexible, dynamic, and programmable functionality of network systems by decoupling control logic from its underlying switches, and introducing a separate control entity to manage and control the network via programming. However, these advantages come with non-negligible penalty to essential network performance such as packet forwarding delay, due to the involvement of logically centralized controllers. Until now, there are a variety of different OpenFlow controllers including NOX, NOX-MT, POX, ONOS, Beacon, Maestro, Ryu, Floodlight, OpenDaylight. These controllers have their own performance characteristics as a result of their design with different technical architecture, implementations in diverse languages and developments by various vendors/research groups. Several measurement tools such as Cbench [16], hcprobe [17], OFsuite_Performance, have been developed to evaluate controller performance with different metrics including latency, throughput, scalability [18]. However, it still needs to spend considerable time for simulation studies or expensive experimental platform setups with these tools. In contrast, analytical models typically based on queueing theory can provide closed-form descriptions of a networking architecture in a few seconds, and have been widely applied to characterize SDN performance.

A few literatures focused on queueing models of controllers as a performance bottleneck in the SDN paradigm. Zuo et al. [10] modeled SDN control plane as $M^m/M/1/B$ by introducing multiple arrivals and single departure queueing system, and derived the queueing delay of flow setup requests in the controller under different network size and traffic load levels. Similarly, Yao et al. [11] also characterized flow setup requests at controllers as batch arrival process, and achieved the $M^k/M/1$ model to estimate average flow service time. However, both models did not reach precise delay estimation, since the batch arrival process could not accurately characterize the pattern of flow setup requests from multiple switches. Wang et al. [19] modeled multiple SDN controllers' performance by using queuing theory, and found that multiple controllers' approach is an effective way to increase the control plane's scalability of SDN, but also results in longer sojourn time. Meanwhile, Fu et al. developed [20] a dormant multi-controller model for centralized multi-controller architecture, i.e., $M/M/c$ queue with N policy and (d, c) vacation mechanism, which allows a part of idle controllers to enter the dormant state under light traffic. Moreover, they established an expected total cost function with key queueing performance metrics, to find the optimal values of various parameters including the number of controllers. However, their cost function did not take into account the expenditure of the controllers. Subsequently, we built an optimization model of controller cluster deployments in consideration of their expenditure to solve the optimal number of controllers in the cluster, by modeling the packet-in message processing of a controller cluster as an $M/M/n$ queue [21].

Some researchers attempted to build precise queueing models of OpenFlow switches. Metter et al. [12,13] formulated a simple analytical model based on $M/M/\infty$ queueing system to understand the impact of flow time-out period on flow table occupancy and controller signaling traffic for different network traffic characteristics. Sood et al. [22] exploited the capabilities of the $M/Geo/1$ queueing model to analyze key performance factors of a SDN switch without the interaction of its controller, including flow-table size, packet arrival rate, number and position of rules. These queueing models were lack of consideration on the priorities of packet flows and Quality of Service (QoS). To address this problem, Miao et al. [23] presented a preemption-based packet scheduling scheme for SDN data plane, and developed an analytical model to achieve its quantitative performance evaluation and pinpoint the performance bottleneck in SDN architecture. As a further step, they characterized bursty traffic generated by multimedia applications as Markov-Modulated Poisson Process (MMPP), and modeled SDN data plane as a priority-queue system to capture the multi-queue nature of forwarding devices [24]. The priority-queue system is decomposed into two single server single queue models by applying and extending the Empty Buffer Approximation (EBA) method, and their key performance metrics were derived in terms of average latency and network throughput. Similarly, Liu et al. [25,26] proposed a prioritized service

queueing model for SDN data plane, evaluated its QoS with self-similar input traffic, and derived the probability distribution of its queue length by introducing Large Deviation Principle (LDP).

Closely related to our work, Jarschel et al. abstracted OpenFlow-based SDN architecture as a feedback-oriented queueing system, where its controller and each switch are respectively modeled as $M/M/1 - S$ and $M/M/1$ queues [27]. They analyzed the forwarding speed and blocking probabilities of the architecture, and estimated packet sojourn time and packet loss probability in such a system. However, the system was just designed for a simple network scenario, where a controller is responsible for only a single switch in the data plane. Mahmood et al. further proposed an analytical model of OpenFlow network performance by approximating the data plane as an open Jackson network with the controller also modeled as an $M/M/1$ queue [14]. They derived the probability distribution function (PDF) and the cumulative distribution function (CDF) of packet forwarding time for a given path. Similarly, Li et al. adopted Jackson's theorem to model file transmission process in Software Defined Satellite Networks (SDSN), and carried out numerical analysis to evaluate impact factors of file sojourn time [15]. Nevertheless, these models was built on arbitrary hypothesis of queueing parameters for viable derivation of network performance metrics. Xiong et al. built a queueing model of packet forwarding in OpenFlow-based SDN, by modeling the packet forwarding of its switches and the packet-in message processing of its controller respectively as $M^X/M/1$ and $M/G/1$ queues, and solved the closed-form expression of packet sojourn time and its PDF [28]. AlGhadhban et al. presented a delay model of flow setup process with consideration of matching probabilities under proactive and reactive flow setup modes [29]. They modeled both the data-plane device and the southbound channel as $M/M/1$ and the control-plane device as $M/G/1$, and derived the overall system capacity and blocking probability. But all above queueing models were built for small-scale network scenarios with a single controller, which are not applicable to the SD-WAN deployments.

3. Software-defined wide-area networks

The increasing number of Internet users and smart mobile terminals have driven a continuous emergence of data-intensive network applications and services such as Internet television, live streaming, and short video sharing. These applications and services generate massive data transmission across wide-area networks, and put forwards higher demands on the quality of service and upper-level application performance. By introducing the concept of software-defined networking, a WAN will obtain many technical advantages, including transport independence, intelligent path control, application optimization, secure connectivity, automatic configuration and management. Consequently, the emerging paradigm of software-defined WAN will become a significant trend for enterprises to save deployment cost, release human resources, and acquire fast, stable, secure transmission service [5,6].

In the SD-WAN paradigm, logically centralized control plane takes charge of all packet forwarding devices connecting a variety of cloud platforms, data centers, enterprises and their branches to Internet. A single controller tends to become a performance bottleneck in such a large-scale network, which results in a natural consequence of multiple controller solutions [30,31]. Fig. 1 demonstrates a typical SD-WAN deployment scenario. In this scenario, OpenFlow switches connect cloud data centers, enterprise headquarters and their branches to Internet through various kinds of high-speed links including MPLS, Ethernet, leased line, xDSL, SDH, 3G/4G LTE. The controller cluster provides the function of topology management, path computation, data security and policy control over global networks.

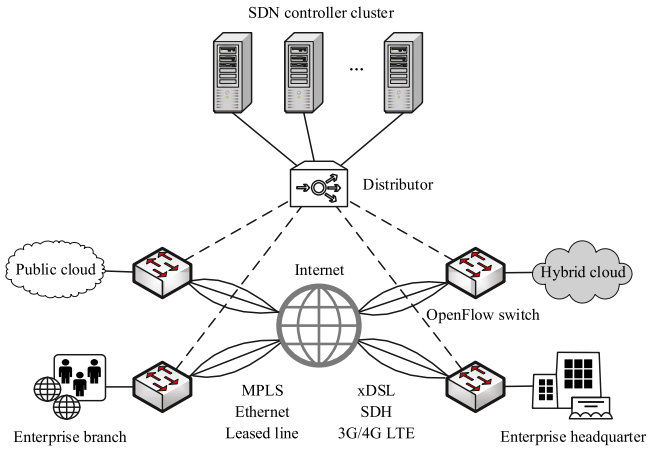


Fig. 1. A typical SD-WAN deployment scenario.

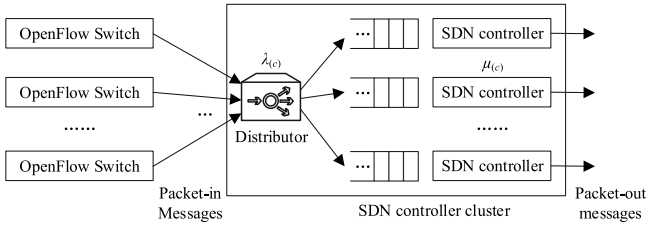


Fig. 2. The packet-in message processing of the controller cluster.

In the above SD-WAN scenario, OpenFlow switches simply forward packets in terms of their flow table entries decided by the global controller cluster. As for an arrived packet, an OpenFlow switch parses it to extract all necessary header fields at each protocol layer, and computes its flow identifier with them. Subsequently, the flow identifier is utilized to match against flow tables to locate an entry. If an entry is successfully matched, its actions will be applied to the packet, generally forwarded to the next station. Otherwise, the packet is supposed to belong to an emerging flow. In such case, the switch should deliver a flow setup request in encapsulation of the packet partly or wholly, i.e., a packet-in message, to the controller cluster for instructions. The controller cluster generates the respective flow rule based on global network view, and installs it to the switch. After that, the switch will process all packets within the flow in terms of the rule.

4. Queueing model of SDN controller cluster

Network data transmission is primarily manifested as a sequence of packet flows in network traffic. As depicted in the previous section, an OpenFlow switch will transmit a packet-in message to the controller cluster for each new flow. Then there will be a packet-in message stream sending from an OpenFlow switch. Packet-in messages from all switches will converge at the controller cluster, and be distributed to multiple controllers for processing in terms of their arrival time. In particular, each controller determines a flow rule for each packet-in message through path computation based on global network topology. The topology is generally built by the running of routing protocols or manual programming. Finally, the controller sends down the flow rule as a packet-out message to all switches among the flow path. Fig. 2 summarizes the packet-in message processing of the controller cluster.

In the SD-WAN paradigm, a packet-in message from an OpenFlow switch is triggered by a packet initiating a new flow. Thus, packet-in messages have a mapping relationship with packet flows at a switch. Meanwhile, a packet belonging to a new flow does not necessarily give rise to a packet-in message from the switch, as the controller cluster may proactively install flow rules into the switch. This is to say, an OpenFlow switch does not need to send a packet-in message for an emerging flow, if the corresponding flow rule has been installed by the controller cluster in advance. In summary, the stream of packet-in messages is mapped with a subset of flow arrival process for an OpenFlow switch.

Network traffic measurements have indicated that packet flows in large-scale networks like WAN tends to be independent of each other and flow arrival process can be approximately regarded as Poisson stream [32–35]. This is attributed to traffic aggregation from a large number of widespread sources in WAN [36,37], which weakens the correlation between packet flows to be negligible. With simple assumption of random mapping between packet-in message stream and flow arrival process, packet-in messages from a switch is supposed to follow Poisson distribution. Suppose packet-in message streams from multiple switches are independent from each other, we can further reach the conclusion that all packet-in messages at the controller cluster form up a Poisson stream in Theorem 4.1.

Theorem 4.1. *As for N OpenFlow switches under the administration of a SDN controller cluster, suppose packet flows independently arrive at the ith ($i = 1, 2, \ldots, N$) one as a Poisson stream with the rate $\lambda_i^{(f)}$, if the α_i part of their corresponding flow rules are randomly installed by the controller cluster in a proactive way, then all packet-in messages at the controller cluster conform to Poisson distribution with the rate $\lambda_{(c)}$:*

$$\lambda_{(c)} = \sum_{i=1}^{N} (1 - \alpha_i)\lambda_i^{(f)}. \tag{1}$$

Proof. Let f_{ij} and m_{ij} ($i = 1, 2, \ldots, N; j = 1, 2, \ldots$) respectively be the jth packet flow passing through the ith switch and the jth packet-in message arriving at the controller cluster from the ith switch. As for each emerging flow, the switch looks up its flow tables to locate an entry. If the corresponding flow entry has been proactively installed by the controller cluster, the lookup will still succeed and the actions in the located entry will be applied to all packets within the flow. Otherwise, the switch will be triggered to fire off a packet-in message m_{ij} to the controller.

According to the assumptions, the flow arrival process $\{f_{ij}\}$ ($j = 1, 2, \ldots$) at the ith switch is supposed as a Poisson stream with the rate $\lambda_i^{(f)}$, and the α_i part of their corresponding flow rules are proactively installed by the controller cluster in a random way. Then it can be concluded that the packet-in message stream $\{m_{ij}\}$ ($j = 1, 2, \ldots$) from the ith switch also conforms to Poisson distribution in (2):

$$\{m_{ij}\} \sim P\left((1 - \alpha_i)\lambda_i^{(f)}\right). \tag{2}$$

With the additivity property of Poisson streams and the simple assumption that packet-in message streams from all switches are independent of each other, we can further achieve the conclusion that all packet-in messages at the controller cluster from N switches conform to Poisson distribution in (3):

$$\left\{\sum_{i=1}^{N} m_{ij}\right\} \sim P\left(\sum_{i=1}^{N} (1 - \alpha_i)\lambda_i^{(f)}\right). \tag{3}$$

Thus, the above theorem is proved.

According to the above queueing analysis, we can characterize the packet-in message processing of the controller cluster as the queueing model $M/M/n$ with the following assumptions: (a) packet-in messages arrive at the controller cluster as a Poisson stream with the average rate $\lambda_{(c)}$; (b) the cluster involves n identical SDN controllers independently processing packet-in messages with the average rate $\mu_{(c)}$; (c) the packet-in message processing time of each controller conforms to negative exponential distribution. According to queueing theory, we can achieve the average sojourn time of packet-in messages in the controller cluster as:

$$\overline{W^{(c)}}(n) = \int_0^{\infty} t w_{(c)}(t) dt = \frac{p_0 \rho_1^n}{\mu_{(c)} n \cdot n! (1 - \rho_{(c)})^2} + \frac{1}{\mu_{(c)}}, \tag{4}$$

where $\rho_1 = \lambda_{(c)}/\mu_{(c)}$, $\rho_{(c)} = \lambda_{(c)}/n\mu_{(c)} < 1$, and p_0 in (5):

$$p_0 = \left(\sum_{i=0}^{n-1} \frac{\rho_1^i}{i!} + \frac{\rho_1^n}{n!} \frac{1}{1 - \rho_{(c)}}\right)^{-1}. \tag{5}$$

5. Queueing model of openflow switches

Packet inter-arrival time has been reported to be exponential and independent in Internet backbone traffic [37–40], although it is generally considered as heavy-tailed distributions in other networks such as access networks and data center networks. This primarily attributes to the fact that packet traffic at the point close to users, terminals and servers is highly correlated due to its dependence on their instantaneous network activities and behaviors, while that far away from these network endpoints tends to be independent as a result of its highly distributed abundant sources. This phenomenon has been formulated as the superposition theorem of point arrival processes [36], which pushes hybrid packet arrivals toward Poisson with the increasing traffic aggregation. Consequently, it can be inferred that packet arrival process at an OpenFlow switch conforms to Poisson distribution in software-defined WAN.

Fig. 3 demonstrates the packet forwarding of an OpenFlow switch. As for an arrived packet, the switch caches it in the ingress queue and sends packets one by one into the processing pipeline of the ingress port. In particular, the switch first parses each packet to extract its key fields and compute its flow identifier. Then the flow identifier is utilized to look up the flow tables to match an entry. If the lookup fails, the switch transmits a packet-in message containing packet information to its superordinate controller, and waits for a corresponding flue rule. Once receiving the flow rule, the switch adds it as a new entry into its flow tables. When a flow entry is found, the packet is switched to the egress port indicated in the entry after applying ingress ACL and updating ingress meters and counters. Note that the packet can be scheduled or queued before entering into the processing pipeline of the egress port. In the egress pipeline, the packet should be first parsed and modified such as its data-link layer header. Then the switch applies egress ACL to the packet, and updates egress meters and counters. Finally the packet is put into the egress queue to wait for transmitting.

As seen from Fig. 3, the packet switching process in an OpenFlow switch can be broken down into a sequence of processing steps. Suppose all s steps are independent of each other and the packet processing time of the ith ($1 \le i \le s$) step conforms to negative exponential distribution with the rate μ_i. Then we can achieve the conclusion that total packet switching time follows a general distribution with the rate in (6) and the variance in (7) according to the Theorem 5.1. In particular, the general distribution is specialized as Erlang if all the steps of the packet switching have identical processing rates.

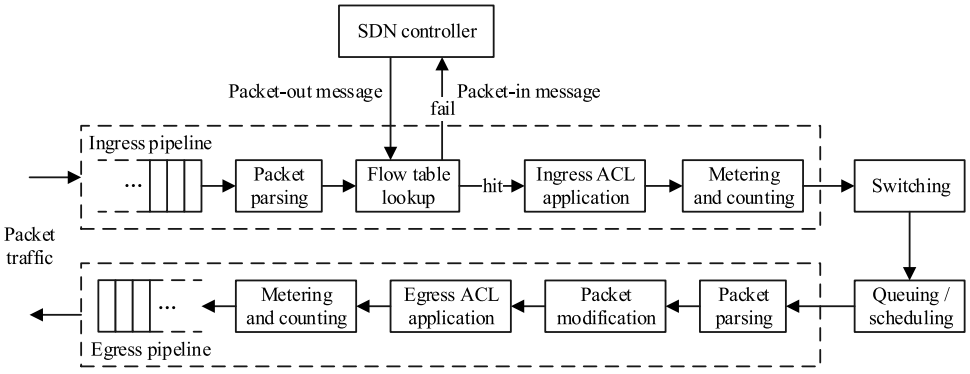


Fig. 3. The packet forwarding of an OpenFlow switch.

Theorem 5.1. *Suppose OpenFlow-based packet switching consists of s processing steps independent of each other, if the processing time of the ith* $(1\le i\le s)$ *step follows negative exponential distribution with the rate* μ_i*, then packet switching time has the rate* $\mu^{(s)}$ *in (6) and the variance* $\sigma^{(s)^2}$ *in (7):*

$$\mu^{(s)} = \frac{1}{\sum_{i=1}^{s} \frac{1}{\mu_i}}. \tag{6}$$

$$\sigma^{(s)^2} = \sum_{i=1}^{s} \frac{1}{\mu_i^2}. \tag{7}$$

Proof. As packet switching is composed of s processing steps, let $T_i(1\le i\le s)$ be the processing time of the ith step, then packet switching time can be expressed as $T = \sum_{i=1}^{s} T_i$. Since T_i follows negative exponential distribution with the rate μ_i according to the assumptions, we can get the expectation and the variance of T_i respectively as $E(T_i) = 1/\mu_i$ and $D(T_i) = 1/\mu_i^2$.

Owing to the assumption that all the steps are independent of each other, we can derive the expectation of the packet switching time T:

$$E(T) = \sum_{i=1}^{s} E(T_i) = \sum_{i=1}^{s} \frac{1}{\mu_i}. \tag{8}$$

Subsequently, we can reach the expected rate of packet switching in (9):

$$\mu^{(s)} = 1/E(T) = \frac{1}{\sum_{i=1}^{s} \frac{1}{\mu_i}}. \tag{9}$$

With the independent assumption of the processing time of all the steps $T_i(1\le i\le s)$, we can derive the variance of packet switching time in (10):

$$\sigma^{(s)^2} = D(T) = \sum_{i=1}^{s} D(T_i) = \sum_{i=1}^{s} \frac{1}{\mu_i^2}. \tag{10}$$

In summary, the theorem is proved.

With the above queueing analysis, we can characterize the packet switching of the ith switch as the queueing model $M/G/1$ based on the following assumptions: (a) packet traffic arrives at the ith switch as Poisson stream with the rate $\lambda_i^{(s)}$; (b) packet switching time of the ith switch conforms to a general distribution with the rate $\mu_i^{(s)}$ in (11) and the variance $\sigma_i^{(s)^2}$ in (12), where the μ_{ij} represents the processing rate of the jth step in the ith switch; (c) each switch processes all arrived packets at each port with the first-come-first-serving principle, and the arrivals and switching of packets are independent of each other.

$$\mu_i^{(s)} = \frac{1}{\sum_{j=1}^{s} \frac{1}{\mu_{ij}}}. \tag{11}$$

$$\sigma_i^{(s)^2} = \sum_{j=1}^{s} \frac{1}{\mu_{ij}^2}. \tag{12}$$

According to queueing theory, we can achieve the average sojourn time of packets in the ith switch in (13):

$$\overline{W_i^{(s)}} = \frac{\overline{L_i^{(s)}}}{\lambda_i^{(s)}} = \frac{1}{\mu_i^{(s)}} + \frac{\rho_i^{(s)^2} + \lambda_i^{(s)^2}\sigma_i^{(s)^2}}{2\lambda_i^{(s)}(1-\rho_i^{(s)})}, \tag{13}$$

where $\rho_i^{(s)} = \lambda_i^{(s)}/\mu_i^{(s)}$.

6. Optimization model of SD-WAN deployments

6.1. Queueing system of packet forwarding in SD-WAN

In software-defined WAN, an OpenFlow switch caches all arrived packets in a queue at each ingress port, and processes them in term of their respective flow rules in flow tables. As for packets with matched entries in the flow tables, the switch directly handles them, generally forwarding them to their next stations. As for packets belonging to new flows, the switch cannot match any entry in the flow tables, and sends packet-in messages containing packet information as flow setup requests to SDN controller cluster. The cluster also keeps all packet-in messages from its subordinate switches in a queue, determines their respective flow rules and delivers the rules as packet-out messages to respective switches. Each switch adds the rules into the flow tables for guiding the processing of subsequent packets within these flows. According to the above queueing analysis in Sections 4 and 5, we can model packet forwarding in software-defined WAN as a queueing system in Fig. 4.

As shown in Fig. 4, packet traffic arrives at the ith $(1\le i\le N)$ switch with the rate $\lambda_i^{(s)}$, and the switch processes packets with the rate $\mu_i^{(s)}$. Suppose packets at the ith switch belong to new flows with the probability q_i, packet flows arrive at the switch with the rate $\lambda_i^{(f)} = q_i\lambda_i^{(s)}$. According to the proof of Theorem 4.1, the switch sends packet-in messages to the controller cluster with the rate $\lambda_i^{(m)} = (1-\alpha_i)\lambda_i^{(f)}$. In summary, the controller cluster receives packet-in messages with the rate $\lambda_{(c)}$ as:

$$\lambda_{(c)} = \sum_{i=1}^{N} \lambda_i^{(m)} = \sum_{i=1}^{N} (1-\alpha_i) q_i \lambda_i^{(s)}. \tag{14}$$

Each controller in the cluster processes packet-in messages as flow setup requests with the rate $\mu_{(c)}$, and the cluster sends packet-out messages in encapsulation of flow rules back to the switch with the rate $\lambda_i^{(m)}$. The switch installs the flow rules in packet-out messages into flow tables, and handles packets within newly installed flows in accordance with the flow rules.

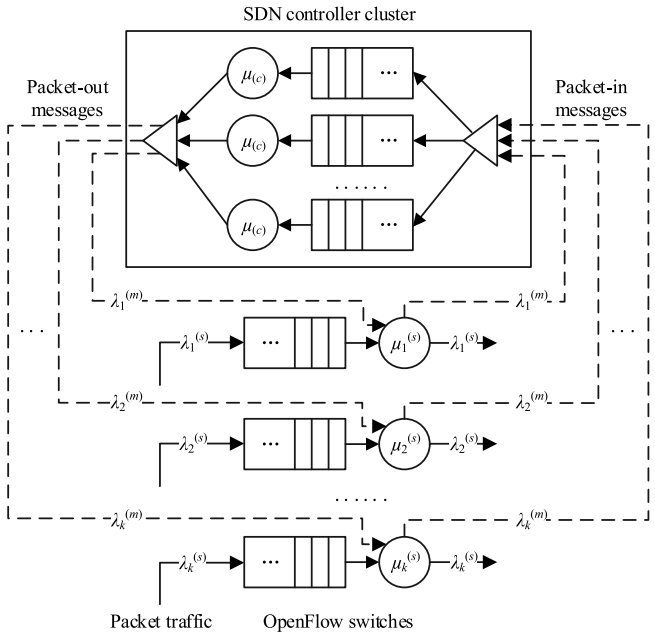


Fig. 4. The queueing system of packet forwarding in software-defined WAN.

6.2. Packet forwarding delay model

With the above queueing system, we can deduce packet forwarding delay at an OpenFlow switch in software-defined WAN. In SD-WAN paradigm, a packet at a switch can be forwarded directly in terms of its flow rule in internal flow tables or indirectly with the involvement of the controller cluster. Indirect forwarding delay at the ith switch consists of direct forwarding delay at the switch and flow setup delay across the controller cluster, respectively denoted as $D_i^{(p)}$ and $D_i^{(f)}(n)$. According to the above queueing model, we can get the probability of packet forwarding at the ith switch with a detour through the controller cluster as $\lambda_i^{(m)}/\lambda_i^{(s)} = (1-\alpha_i)q_i$. Consequently, we can express packet forwarding delay at the ith switch $D_i(n)$ $(1 \le i \le N)$ in (15):

$$D_i(n) = \begin{cases} D_i^{(p)} & \text{with probability } 1-(1-\alpha_i)q_i, \\ D_i^{(f)}(n) + D_i^{(p)} & \text{with probability } (1-\alpha_i)q_i. \end{cases} \tag{15}$$

With the above packet forwarding delay at the ith switch, we achieve its expectation $\overline{D_i}(n)$ in (16):

$$\overline{D_i}(n) = (1-\alpha_i)q_i\overline{D_i^{(f)}}(n) + \overline{D_i^{(p)}}, \tag{16}$$

where $\overline{D_i^{(f)}}(n)$ and $\overline{D_i^{(p)}}$ respectively denotes average flow setup delay and the average delay of direct packet forwarding at the ith switch. Meanwhile, we can also get the expectation of maximal packet forwarding delay at the ith switch $\widehat{D}_i(n)$ in (17):

$$\widehat{D}_i(n) = \overline{D_i^{(f)}}(n) + \overline{D_i^{(p)}}. \tag{17}$$

Since packet arrives at the ith switch with the rate $\lambda_i^{(s)}$, we further compute average packet forwarding delay among the entire network $\overline{D}(n)$ in (18):

$$\begin{aligned} \overline{D}(n) &= \frac{\sum_{i=1}^{N}\lambda_i^{(s)}\overline{D_i}(n)}{\sum_{i=1}^{N}\lambda_i^{(s)}} \\ &= \frac{\sum_{i=1}^{N}(1-\alpha_i)q_i\lambda_i^{(s)}\overline{D_i^{(f)}}(n)}{\sum_{i=1}^{N}\lambda_i^{(s)}} + \frac{\sum_{i=1}^{N}\lambda_i^{(s)}\overline{D_i^{(p)}}}{\sum_{i=1}^{N}\lambda_i^{(s)}}. \end{aligned} \tag{18}$$

Delay in packet switching networks can be divided into four parts: transmission delay, propagation delay, queueing delay and processing delay. Note that the transmission and propagation delays in software-defined WAN are negligible compared to the queueing and processing delays, since the transmission rates come up to 1Gbps for modern switches and controller clusters, and the propagation rate of electromagnetic wave goes beyond 2.0×10^8 m/s typically in copper wire and optical fiber. Then we only need to focus on the queueing and processing delays collectively called sojourn time at switches and the controller cluster.

Therefore, the average delay of direct packet forwarding $\overline{D_i^{(p)}}$ can be approximately estimated as the average packet sojourn time $\overline{W_i^{(s)}}$ at the ith switch. As for the average flow setup delay $\overline{D_i^{(f)}}(n)$ at the ith switch, we can infer it from three flow setup phases respectively, i.e., a packet-in message from the switch to controller cluster, flow rule decision in the controller cluster, and a packet-out message from the controller cluster to the switch. Suppose the controller cluster is connected to the cth switch, and the messages are transmitted between the ith switch and it through the path $P(i, c)$ exclusive of the i switch. With the average message sojourn time at the controller cluster $\overline{W^{(c)}}(n)$ and the average packet sojourn time at the ith switch $\overline{W_i^{(s)}}$, we get the average flow setup delay $\overline{D_i^{(f)}}(n)$ in (19):

$$\overline{D_i^{(f)}}(n) = \overline{W^{(c)}}(n) + \overline{W_i^{(s)}} + \sum_{k\in P(i,c)} 2\overline{W_k^{(s)}}. \tag{19}$$

With the average message sojourn time at the controller cluster in (4), we further deduce the average packet forwarding delay $\overline{D}(n)$ in (20) from (18):

$$\begin{aligned} \overline{D}(n) &= a_1\overline{W^{(c)}}(n) + a_{00} + a_{01}, \\ &= \frac{a_1 p_0 \rho_1^n}{\mu_{(c)}(1-\rho_{(c)})^2 n\cdot n!} + \frac{a_1}{\mu_{(c)}} + a_{00} + a_{01}. \end{aligned} \tag{20}$$

where,

$$\begin{aligned} a_1 &= \frac{\sum_{i=1}^{N}(1-\alpha_i)q_i\lambda_i^{(s)}}{\sum_{i=1}^{N}\lambda_i^{(s)}}, \\ a_{00} &= \frac{\sum_{i=1}^{N}[1+(1-\alpha_i)q_i]\lambda_i^{(s)}\overline{W_i^{(s)}}}{\sum_{i=1}^{N}\lambda_i^{(s)}}, \\ a_{01} &= \frac{\sum_{i=1}^{N}\sum_{k\in P(i,c)}2(1-\alpha_i)q_i\lambda_i^{(s)}\overline{W_k^{(s)}}}{\sum_{i=1}^{N}\lambda_i^{(s)}}. \end{aligned} \tag{21}$$

Similarly, we achieve the expectation of maximal packet forwarding delay at the ith switch $\widehat{D}_i(n)$ in (22) from (17):

$$\widehat{D}_i(n) = \overline{W^{(c)}}(n) + b_i = \frac{p_0\rho_1^n}{\mu_{(c)}(1-\rho_{(c)})^2 n\cdot n!} + \frac{1}{\mu_{(c)}} + b_i, \tag{22}$$

where,

$$b_i = 2\overline{W_i^{(s)}} + \sum_{k\in P(i,c)} 2\overline{W_k^{(s)}}. \tag{23}$$

As seen from (20) and (22), the number of controllers in the cluster n has a significant impact on packet forwarding delays. This is greatly attributed to the fact that the controller cluster operates at a logically-centralized mode.

6.3. Optimization model of controller cluster deployments

In software-defined WAN, upper-layer applications have varying levels of requirements on quality of service (QoS), primarily manifested as packet forwarding delay. As shown in the above section, both the average packet forwarding delay $\overline{D}(n)$ and the expectation of maximal packet forwarding delay at the ith switch

$\widehat{D}_i(n)$ are significantly dominated by the number of controllers in the cluster n. Therefore, it is requisite to determine the optimal number of controllers in a cluster for the deployments of software-defined WAN.

As seen from (20), the average packet forwarding delay gradually decreases with the increasing number of controllers. This, in turn, would need more expenditure to purchase, maintain, or upgrade controllers. For simplicity, we formulate the capital expenditure of the controller cluster as $e(n) = e_1 n + e_0$, where e_1 and e_0 respectively the cost of each controller and the other cost of the cluster except of its controllers. In summary, we build a multi-objective optimization model of controller cluster deployments in (24):

$$\begin{aligned} &\underset{n}{argmin}\big(\overline{D}(n), e(n)\big) \\ &\text{subject to} \\ &\quad \widehat{D}_i(n) \le D_{QoS},\ 1 \le i \le N, \\ &\quad e(n) \le B_{CC}, \\ &\quad n \in N_+, \end{aligned} \tag{24}$$

by taking the average packet forwarding delay and the capital expenditure of the controller cluster as the optimization objectives.

As shown in (24), the optimization model has three constraint conditions: (a) maximal packet forwarding delay at each switch must not exceed the upper bound allowed by the quality of service of network applications D_{QoS}; (b) there is a budget ceiling B_{CC} for the capital expenditure of the controller cluster; (c) the number of controllers n is naturally a positive integer.

To solve the above model, we convert its multi-objective optimization function to a single-objective one, by multiplying the two optimization objectives $\overline{D}(n)$ and $e(n)$. However, there is a non-elementary factor $n!$ in the formula of $\overline{D}(n)$ shown in (20). This makes it hard to solve the minimum point of the single-objective optimization function $f(n)$. Fortunately, $n!$ can be approximated as $\sqrt{2\pi n}(e/n)^n$ by Stirling's formula. Therefore, the optimization function $f(n)$ can be inferred as an elementary function in (25):

$$\begin{aligned} f(n) &= e(n) \times \overline{D}(n) \\ &= (e_1 n + e_0) \times \left[\frac{a_1 p_0 \rho_1^n}{\mu_{(c)}(1-\rho_{(c)})^2 n \cdot n!} + \frac{a_1}{\mu_{(c)}} + a_0 \right] \\ &\approx (e_1 n + e_0) \times \left[\frac{a_1 p_0 \rho_1^n e^n}{\sqrt{2\pi}\mu_{(c)}(1-\rho_{(c)})^2 n^{n+3/2}} + \frac{a_1}{\mu_{(c)}} + a_0 \right], \end{aligned} \tag{25}$$

where $a_0 = a_{00} + a_{01}$.

Similarly, the first constraint condition in (24) can be converted into an elementary inequation group in (26) by substituting (22) into its left-hand side and applying Stirling's formula.

$$\frac{p_0 \rho_1^n e^n}{\sqrt{2\pi}\mu_{(c)}(1-\rho_{(c)})^2 n^{n+3/2}} + \frac{1}{\mu_{(c)}} + b_i \le D_{QoS},\ 1 \le i \le N. \tag{26}$$

Then the constraint condition can be further simplified as a single elementary inequation in (27):

$$\frac{p_0 \rho_1^n e^n}{\sqrt{2\pi}\mu_{(c)}(1-\rho_{(c)})^2 n^{n+3/2}} + \frac{1}{\mu_{(c)}} + \widehat{b} \le D_{QoS}, \tag{27}$$

where $\widehat{b}$ stands for the maximum of b_i in (28):

$$\widehat{b} = \max_{1 \le i \le N}\{b_i\} = \max_{1 \le i \le N}\left\{ 2\overline{W_i^{(s)}} + \sum_{k \in P(i,c)} 2\overline{W_k^{(s)}} \right\}. \tag{28}$$

In summary, the single-objective optimization model can be formalized in (29):

$$\begin{aligned} &\underset{n}{argmin}(e_1 n + e_0) \\ &\qquad \times \left[\frac{a_1 p_0 \rho_1^n e^n}{\sqrt{2\pi}\mu_{(c)}(1-\rho_{(c)})^2 n^{n+3/2}} + \frac{a_1}{\mu_{(c)}} + a_0 \right] \\ &\text{subject to} \\ &\quad \frac{p_0 \rho_1^n e^n}{\sqrt{2\pi}\mu_{(c)}(1-\rho_{(c)})^2 n^{n+3/2}} + \frac{1}{\mu_{(c)}} + \widehat{b} - D_{QoS} \le 0, \\ &\quad e_1 n + e_0 - B_{CC} \le 0, \\ &\quad n \in N_+. \end{aligned} \tag{29}$$

6.4. Optimization model solution

The above model in (29) is an optimization problem with inequality constraints. However, it cannot be solved by applying Lagrange multipliers and Karush-Kuhn-Tucker conditions, due to the complexity of its objective function and first constraint. Fortunately, the objective function can be proved to be convex in Theorem 6.1.

Theorem 6.1. *The objective function in (29) is convex if its all parameters are positive and $\rho_1 < n$.*

Proof. As for the proof of a convex function, it is identical to prove the positive of its second derivative. Suppose $d(n) = h(n)/g(n) + 1$ where $g(n) = \sqrt{2\pi}(1-\rho_{(c)})^2 n^{n+3/2}$ and $h(n) = p_0 \rho_1^n e^n$, we can simplify the objective function in (29) as the expression in (30):

$$f(n) = (e_1 n + e_0) \times \left[\frac{a_1}{\mu_{(c)}} d(n) + a_0 \right]. \tag{30}$$

To solve the derivative of $g(n)$, we rewrite it as $g(n) = \sqrt{2\pi}(1-\rho_{(c)})^2 e^{(n+3/2)\ln n}$. Then, its derivative can be computed in (31) by the derivation rules of exponential, product and logarithmic functions:

$$\begin{aligned} g'(n) &= \sqrt{2\pi}(1-\rho_{(c)})^2 \left(\frac{3}{2n} + \ln n + 1 \right) e^{(n+3/2)\ln n} \\ &= \left(\frac{3}{2n} + \ln n + 1 \right) g(n). \end{aligned} \tag{31}$$

Meanwhile, we can also get the derivative of $h(n)$ in (32) according to the derivation rule of product and exponential functions:

$$h'(n) = p_0(1 + \ln \rho_1)\rho_1^n e^n = (1 + \ln \rho_1) h(n). \tag{32}$$

With the derivatives of $g(n)$ and $h(n)$, we can calculate the derivative of $d(n)$ in (33) in terms of the quotient rule. Note that $\ln(n/\rho_1)$ is positive owing to $\rho_1 < n$. Since all parameters are positive, we can infer the positive of $g(n)$ and $h(n)$. Then, it can be concluded that $d'(n)$ in (33) is negative.

$$d'(n) = \frac{g(n)h'(n) - g'(n)h(n)}{g^2(n)} = -\left(\frac{3}{2n} + \ln \frac{n}{\rho_1} \right) \frac{h(n)}{g(n)}. \tag{33}$$

Then we further achieve the derivative of $f(n)$ in (34) in accordance with the product rule:

$$\begin{aligned} f'(n) &= \frac{a_1(e_1 n + e_0)}{\mu_{(c)}} d'(n) + \frac{a_1 e_1}{\mu_{(c)}} d(n) + a_0 e_1, \\ &= \left[\frac{a_1 e_1}{\mu_{(c)}} - \frac{a_1}{\mu_{(c)}}(e_1 n + e_0)\left(\frac{3}{2n} + \ln \frac{n}{\rho_1} \right) \right] \frac{h(n)}{g(n)} \\ &\quad + \frac{a_1 e_1}{\mu_{(c)}} + a_0 e_1. \end{aligned} \tag{34}$$

For simple expression of $f'(n)$, we assume a function of n in (35):

$$k(n) = \frac{a_1 e_1}{\mu_{(c)}} - \frac{a_1}{\mu_{(c)}}(e_1 n + e_0)\Big(\frac{3}{2n} + \ln\frac{n}{\rho_1}\Big). \tag{35}$$

Then we can compute its derivative in (36) by the derivation rules of product, reciprocal and logarithmic functions:

$$k'(n) = \frac{3a_1 e_0}{2\mu_{(c)} n^2} - \frac{a_1}{\mu_{(c)}} \ln\frac{n}{\rho_1}\Big(2e_1 + \frac{e_0}{n}\Big). \tag{36}$$

With the function $k(n)$, the derivative of $f(n)$ can be simplified as $f'(n) = h(n)k(n)/g(n) + a_1 e_1/\mu_{(c)} + a_0 e_1$. According to the quotient and product rules, we can eventually compute the second derivative of $f(n)$ in (37) with the functions $g(n)$, $h(n)$, $k(n)$ and their derivatives:

$$\begin{aligned} f''(n) &= \frac{g(n)[h(n)k'(n) + h'(n)k(n)] - g'(n)h(n)k(n)}{g^2(n)}, \\ &= \frac{a_1}{\mu_{(c)}}\left[(e_1 n + e_0)\ln^2\frac{n}{\rho_1} + \frac{2e_0}{n}\ln\frac{n}{\rho_1} + \frac{15e_0}{4n^2} + \frac{3e_1}{4n}\right]\frac{h(n)}{g(n)}. \end{aligned} \tag{37}$$

As in (37), $\ln(n/\rho_1)$ is positive owing to $\rho_1 < n$. Since all parameters are positive, we can further infer the positive of the function $g(n)$ and $h(n)$ and all other items in (37). Eventually, we can conclude the positive of $f''(n)$. That is to say, the function $f(n)$ is convex. The theorem is proved.

As for the first constraint in (29), its left-hand side can be expressed as $d(n)/\mu_{(c)} - a_0 + \widehat{b} - D_{QoS}$, where $d(n)$ is defined in the proof of Theorem 6.1 and its first derivative $d'(n)$ is concluded to be negative in (33). Thus, we can infer that the left-hand expression of the first constraint has a negative derivative $d'(n)/\mu_{(c)}$. This implies that the expression decreases monotonically with the increasing of n, and the first constraint regulates the lower bound of n represented as n_{low}. Meanwhile, the second constraint in (29) regulates the upper bound of n as $n_{up} = (B_{CC} - e_0)/e_1$. With the third constraint that n must be a positive integer, all three constraints in (29) can be summarized as that n should fall into an integer range $[n_{low}, n_{up}]$.

Consequently, we can solve the optimization model in (29) by searching in the integer range $[n_{low}, n_{up}]$ for the optimal value of n with the minimum of the convex objective function $f(n)$. Table 1 summarizes the solution algorithm of the optimization model in (29) for the deployments of software-defined WAN. The algorithm primarily consists of three parts: (a) calculating intermediate parameters in *Line 1–11*; (b) determining the upper and lower bounds of n in *Line 12–21*; (c) looking for the optimal value of n in *Line 22–32*.

Firstly, we calculate all necessary intermediate parameters, including packet switching delay of each switch in *Line 5*, and the arrival rate of packet-in messages at the controller cluster in *Line 11*. The parameters a_0, a_1 and $\widehat{b}$ in the optimization model are computed in *Line 6–10* based on network topology information. Secondly, the upper bound of n is directly gotten from the budget of the controller cluster in *Line 12*, and the lower bound of n is located by calculating packet-in message processing time of the controller cluster in *Line 16*, and judging whether it satisfies the delay requirement of quality of service in *Line 17*. Note that the processing rate of packet-in messages must be larger than the arrival rate of these at the controller cluster in *Line 14–15*. Thirdly, we seek for the optimal value of n from its lower bound in *Line 24–25*, and stop at a higher value of the objective function in *Line 26–31* owing to the convexity of the objective function.

7. Experiments

This section evaluates our proposed queueing model of controller cluster with the measurement tool OFsuite_Performance, performs numerical analysis on packet forwarding delay, and solves optimal number of controllers under different parameters.

7.1. Queueing model comparison for controller clusters

In our experiments, we utilize the measurement tool OFsuite_Performance developed by SDNCTC (Global SDN Certified Testing Center) to evaluate the performance models of SDN controller cluster. The tool measures OpenFlow message processing capacity of SDN controllers by simulating a variety of network topologies, a large number of OpenFlow 1.3 switches, and all types of OpenFlow events. In particular, all simulated OpenFlow switches are connected to a controller cluster via network interface cards, and send a stream of packet-in messages to the cluster replying with packet-out messages. The tool records the sending time of packet-in messages and the arrival time of corresponding packet-out messages for each switch, and computes the arrival rates and the processing rates of packet-in messages at the controller cluster by inference.

As for SDN controller to be measured, we select the popular open-source controller OpenDaylight supporting clustering. The controller is respectively installed in a single-point mode with 1 server and a cluster mode with 3 servers. Then the measurement tool simulates wide-area networks with the number of switches varying from 9 to 14 for the single-point mode and from 4 to 9 per controller for the cluster mode. Each simulated switch is configured to send packet-in messages to the controller cluster at the rate 1 k/s. We perform iterative testing for 5 times for each network scenario, and get average controller performance parameters in Table 2.

As seen from Table 2, packet-out message rates always keep pace with their corresponding packet-in message rates and average measured delay stays at millisecond levels for packet-in message rates below 12 k/s and 21 k/s respectively for 1 and 3 controller servers. However, packet-out message rates will subsequently cut down and average measured delay will sharply rise to second levels for higher packet-in message rates. Thus we can infer that the single controller and the controller cluster respectively processes packet-in messages at an approximate rate of 12 k/s and 21.5 k/s. Furthermore, it can be noticed that the processing rate of the controller cluster is disproportionate to that of the single controller, probably due to additional overheads of the controller cluster.

Subsequently, we simulate wide-area networks with different number of switches no more than 11 for the single-point mode and 7 per controller for the cluster mode. Then we carry out controller performance evaluation in the same way as above. Fig. 5 illustrates average measured delay of packet-in messages across the single controller and the controller cluster. Meanwhile, we get the estimated delay of our proposed model $M/M/n$ in Fig. 5(b) from (4) and (5), with the number of controllers $n = 3$ in the cluster, the varying arrival rates $\lambda_{(c)}$ and the processing rate $\mu_{(c)} = 21.5$ k/s of packet-in messages at the controller cluster. Similarly, the estimated delay of the traditional model $M^k/M/n$ [11] can be also calculated in Fig. 5(b) with the above performance parameters and the number of switches per controller.

As seen from Fig. 5, our proposed model $M/M/n$ has a closer estimated delays to measured delays than the $M^k/M/n$ model. In particular, all measured delays slightly go beyond the estimated ones of the $M/M/n$ model. This is probably ascribed to the fact that measured delays include message transmission delays

Table 1
The solution algorithm of the optimization model for SD-WAN deployments.

Input: (a) network topology information $G(V, E)$, including all switches, switch inter-connection relationship, and the switch point where controllers are deployed; (b) parameters regarding the i switch in V, including packet arrival rate $\lambda_i^{(s)}$, packet processing rate μ_{ij} of the j step, the probability of a packet belonging to a new flow q_i, and the ratio of proactively installed flow rules α_i; (c) controller cluster information, including packet-in message processing rate of a single controller $\mu_{(c)}$, the cost of each controller e_1, the other cost of the cluster except of its controllers e_0, and the budget ceiling of the controller cluster B_{CC}; (d) maximal packet forwarding delay D_{QoS} allowed by the quality of service.
Output: the optimal value of the controller number n.

for $i \leftarrow 1, |V|$ **do**
  calculate the packet switching rate of the ith switch $\mu_i^{(s)}$ in (11) and the variance ${\sigma_i^{(s)}}^2$ in (12);
  if $\lambda_i^{(s)} \geq \mu_i^{(s)}$, **then**
    exit;
  calculate the average packet switching time of the ith switch $\overline{W_i^{(s)}}$ in (13) with $\lambda_i^{(s)}$, $\mu_i^{(s)}$ and ${\sigma_i^{(s)}}^2$;
calculate the parameter a_{00} in (21) with $\overline{W_i^{(s)}}(1 \leq i \leq |V|)$;
calculate the parameter a_{01} in (21) and b_i in (23) based on network topology $G(V, E)$ with $\overline{W_i^{(s)}}(1 \leq i \leq |V|)$;
$a_0 \leftarrow a_{00} + a_{01}$;
calculate the parameters a_1 in (21) with parameters regarding all switches;
$\widehat{b} \leftarrow \max_{1 \leq i \leq |V|} \{b_i\}$;
calculate packet-in message arrival rate of the controller cluster $\lambda_{(c)}$ in (14);
$n_{low} \leftarrow \lceil \lambda_{(c)} / \mu_{(c)} \rceil$;
$n_{up} \leftarrow \lfloor (B_{CC} - e_0)/e_1 \rfloor$;
for $i \leftarrow 1, n_{up}$ **do**
  calculate the average sojourn time of packet-in messages in the controller cluster $\overline{W^{(c)}}(i)$ in (4);
  if $\overline{W^{(c)}}(i) + \widehat{b} \leq D_{QoS}$, **then**
    break;
if $i > n_{up}$, **then**
  exit;
$n_{low} \leftarrow i$;
if $n_{low} = n_{up}$, **then**
  return n_{low};
calculate the value of objective function $f(n_{low})$ in (25) with a_0, a_1, e_0, e_1 and other parameters;
$f_{min} \leftarrow f(n_{low})$;
for $i \leftarrow n_{low} + 1, n_{up}$ **do**
  calculate the value of objective function $f(i)$ in (25);
  if $f_{min} > f(i)$, **then**
    $f_{min} \leftarrow f(i)$;
  else
    break;
return $i - 1$;

Table 2
The controller performance parameters.

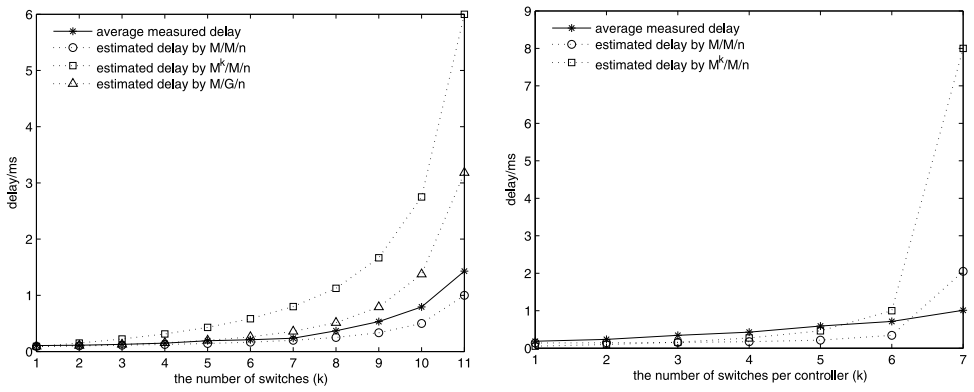
(a) 1 controller						
Performance parameters	The number of switches					
	9	10	11	12	13	14
Packet-in message rates	9000	10000	11000	12000	13000	14000
Packet-out message rates	9000	10000	11000	12000	11882	11695
Average measured delay (ms)	0.533	0.794	1.43	3.38	62.54	197.75
(b) 3 controllers						
Performance parameters	The number of switches per controller					
	4	5	6	7	8	9
Packet-in message rates	12000	15000	18000	21000	24000	27000
Packet-out message rates	12000	15000	18000	21000	20166	18966
Average measured delay (ms)	0.428	0.592	0.715	1.013	192.93	349.45

and propagation delays, besides of the sojourn time of packet-in messages in controller cluster. Furthermore, estimated delays of the $M^k/M/n$ model become much greater than measured delays with the increasing number of switches. This phenomenon is attributed to the characteristics of the $M^k/M/n$ model that its estimated delays have a strong positive correlation with the number of packet-in messages in a batch, i.e., the number of OpenFlow switches.

7.2. Packet forwarding delay

As for a SD-WAN scenario, we numerically analyze the impact of different parameters on packet forwarding delay. To simplify the numerical analysis, the network scenario is supposed to be a complete tree topology with the degree $m = 10$, and the root switch connects a controller cluster. Then we can compute average and maximum distance between a switch and the controller cluster, given the number of switches. In addition, there are 10 processing steps for packet switching shown in Fig. 3. With simple assumption of all the packet processing steps with identical rate in a switch, we can get the relationship between the packet switching rate and its standard variance as $\sigma_i^{(s)} = \frac{1}{\sqrt{10}\mu_i^{(s)}}$.

All other parameters are typically configured as follows. There are 100 OpenFlow switches with packet switching rate $\mu_i^{(s)} = 60$ k/s. Each switch receives packets with the rate $\lambda_i^{(s)} = 20$ k/s, $q_i = 0.04$ of which belong to new flows. Among all flow rules in each switch, the controller cluster installs $\alpha_i = 0.8$ of them in a proactive mode. Each controller in the cluster processes packet-in



(a) 1 controller (b) 3 controllers

Fig. 5. The estimated delay comparison of controller performance models.

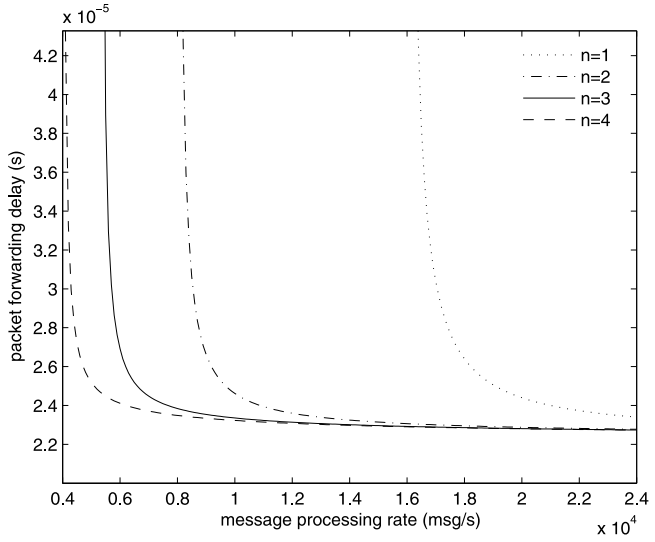


Fig. 6. The relationship of packet forwarding delay and packet-in message processing rate of the controller cluster.

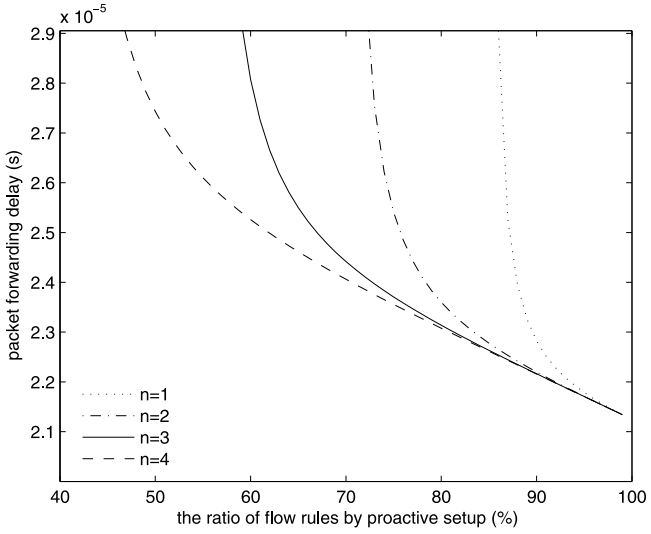


Fig. 7. The relationship of packet forwarding delay and the ratio of proactively installed flow rules.

messages at the rate $\mu_{(c)} = 12$ k/s. The above parameter configurations are taken as the benchmark of our following numerical analysis.

With the increasing packet-in message processing rate of a controller $\mu_{(c)}$, we can calculate average packet forwarding delay $\overline{D}(n)$ for different number of controllers in the cluster n in Fig. 6. As shown in Fig. 6, the higher processing rate of a controller, the lower forwarding delay for identical number of controllers. In particular, the forwarding delay will become stable nearly to 2.3×10^{-5} s when the processing rate rise up adequately. This lower bound of the forwarding delay is primarily generated in packet queueing and processing on switches. Furthermore, we can see from Fig. 6 that a controller should be able to process packet-in messages at the minimal rate 16 k/s, 8 k/s, 5.3 k/s, 4 k/s respectively for 1, 2, 3, 4 controllers. That is to say, the controller cluster should be capable of processing packet-in messages with the rate 60 k/s.

With the increasing ratio of flow rules proactively installed by the controller cluster α_i, we can calculate average packet forwarding delay $\overline{D}(n)$ for different number of controllers in the cluster n in Fig. 7. As shown in Fig. 7, the higher ratio of proactively installed flow rules, the lower forwarding delay for identical number of controllers. In particular, the forwarding delay is always higher than 2.1×10^{-5} s even if the ratio of proactively installed flow rules goes up to 100%. The lower bound of forwarding delay can be approximated as the total packet delay on switches. Moreover, we can see from Fig. 7 that the controller cluster should proactively install flow rules with a minimum of 85%, 70%, 55%, 40% respectively for 1, 2, 3, 4 controllers. This is to say, each controller can only manage 15% of packet-in messages from all switches.

With the increasing number of OpenFlow switches N, we can calculate average packet forwarding delay $\overline{D}(n)$ for different number of controllers in the cluster n in Fig. 8. As shown in Fig. 8, the larger number of switches, the higher forwarding delay for identical number of controllers. In particular, the forwarding delay will come close to 2.2×10^{-5} s when the number of switches is decreased to 1. This lower bound of forwarding delay is dominated by the packet delay of queueing and processing on switches. In addition, we can see from Fig. 8 that the controller cluster can manage 75, 150, 225, 300 switches at most respectively for 1, 2, 3, 4 controllers. That is to say, each controller can take charge of 75 switches.

With the increasing packet arrival rate of each OpenFlow switch $\lambda_i^{(s)}$, we can calculate average packet forwarding delay $\overline{D}(n)$ for different number of controllers in the cluster n in Fig. 9. As

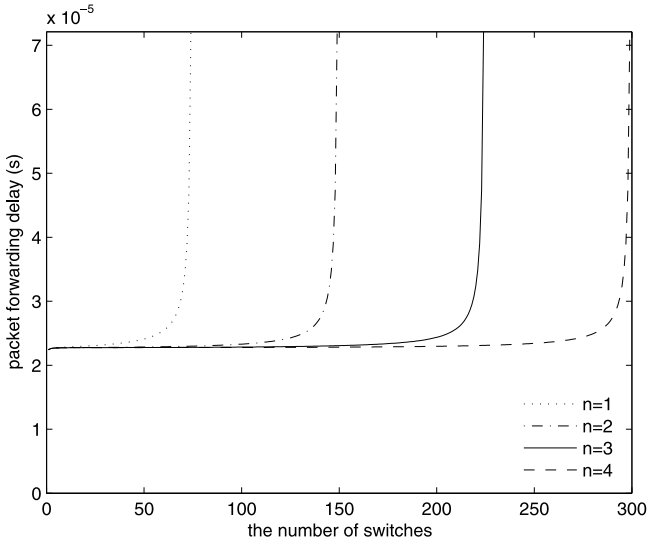


Fig. 8. The relationship of packet forwarding delay and the number of switches.

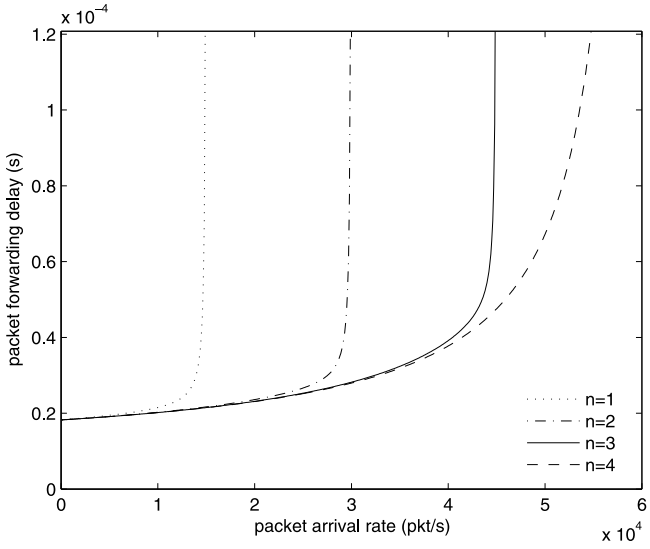


Fig. 9. The relationship of packet forwarding delay and packet arrival rate at each switch.

shown in Fig. 9, the higher packet arrival rate at each switch, the higher forwarding delay for identical number of controllers. In particular, the forwarding delay will go down to 1.8×10^{-5} s when the packet arrival rate comes close to 0. This lower bound can be considered as the minimum of packet forwarding delay in such a network scenario. Besides, we can see from Fig. 9 that the packet arrival rate cannot go beyond 15 k/s, 30 k/s, 45 k/s, 60 k/s respectively for 1, 2, 3, 4 controllers. This is to say, each controller can be responsible for packets with the arrival rate 15 k/s at each switch.

7.3. Optimal number of controllers

The optimal number of controllers is solved by following the above network scenario and essential parameter configurations. Additionally, we set capital expenditure parameters regarding the controller cluster as: the cost of each controller $e_1 = 20$, the other cost of the cluster $e_0 = 20$, and the budget ceiling of the controller cluster $B_{CC} = 180$. This implies that the number of controllers

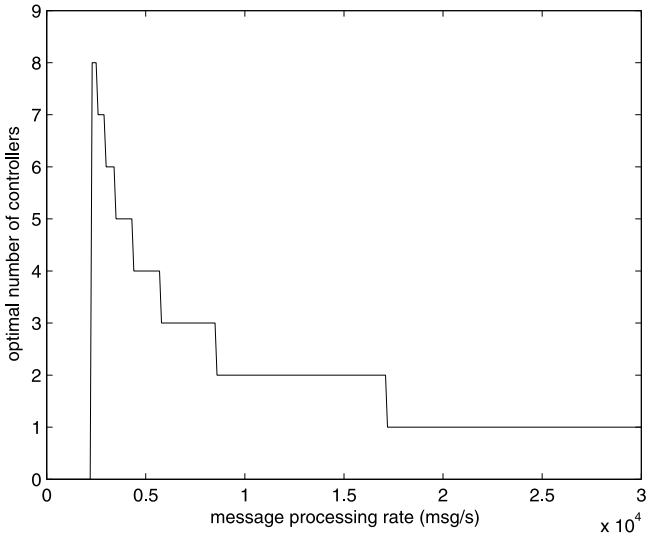


Fig. 10. The optimal number of controllers for increasing packet-in message processing rate of a controller.

is limited with the upper bound 9. Moreover, packet forwarding delay at a switch is confined below $D_{QoS} = 1ms$ by quality of service.

According to the solution algorithm of the optimization model of controller cluster deployments in Table 1, we first solve the optimal number of controllers n for increasing packet-in message processing rate of a controller $\mu_{(c)}$ in Fig. 10. As shown in Fig. 10, the higher processing rate of a controller, the fewer number of controllers. In particular, it is optimal to deploy 1, 2, 3, 4, 5, 6, 7, 8 controllers respectively for packet-in message processing rate of a controller above 17.2 k/s, 8.6 k/s, 5.8 k/s, 4.4 k/s, 3.5 k/s, 3 k/s, 2.6 k/s, 2.3 k/s. Furthermore, there is no solution if a controller processes packet-in messages with the rate below 2.3 k/s. This is attributed to the fact that it will need more controllers to keep low packet forwarding delay demanded by quality-of-service, but will go beyond the budget ceiling of the controller cluster.

Secondly, we solve the optimal number of controllers n with the rate $\mu_{(c)} = 9$ k/s for increasing ratio of flow rules proactively installed by the controller cluster α_i in Fig. 11. As shown in Fig. 11, the higher ratio of proactively installed flow rules, the fewer number of controllers. In particular, it is optimal to deploy 1, 2, 3, 4, 5, 6, 7, 8 controllers respectively for the ratio of proactively installed flow rules above 91%, 79%, 69%, 59%, 48%, 38%, 23%, 12%. Moreover, there is no solution if flow rules are proactively installed with the ratio below 12%. This is attributed to the fact that there will be too many flow setup requests arrived at the controller cluster, which leads to high packet forwarding delay beyond the requirements of quality-of-service.

Thirdly, we solve the optimal number of controllers n for increasing number of OpenFlow switches N in Fig. 12. As shown in Fig. 12, the larger number of switches, the more number of controllers. In particular, it is optimal to deploy 1, 2, 3, 4, 5, 6, 7, 8 controllers respectively for the number of switches no more than 67, 142, 217, 290, 364, 437, 517, 591. In addition, there is no solution for the number of switches in excess of 591. This is attributed to the fact that too many switches will generate excessive packet-in messages, and result in larger packet-in message processing delay at the controller cluster than that regulated by quality-of-service.

Fourthly, we solve the optimal number of controllers n for increasing packet arrival rate of OpenFlow switches $\lambda_i^{(s)}$ in Fig. 13. As shown in Fig. 13, one more controller should be added for a

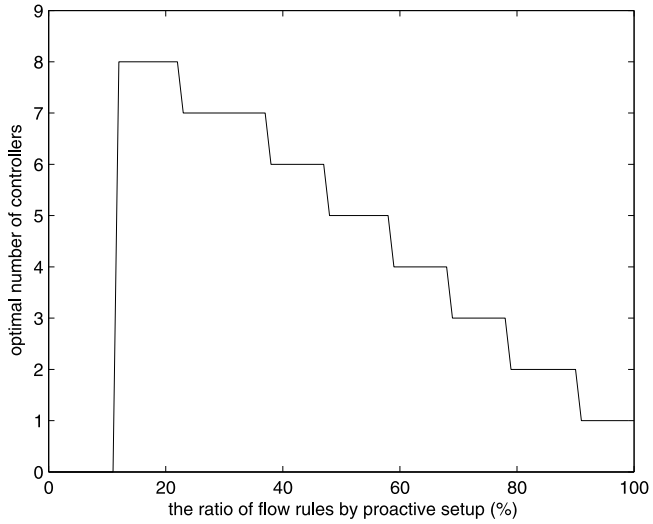


Fig. 11. The optimal number of controllers for increasing ratio of proactively installed flow rules.

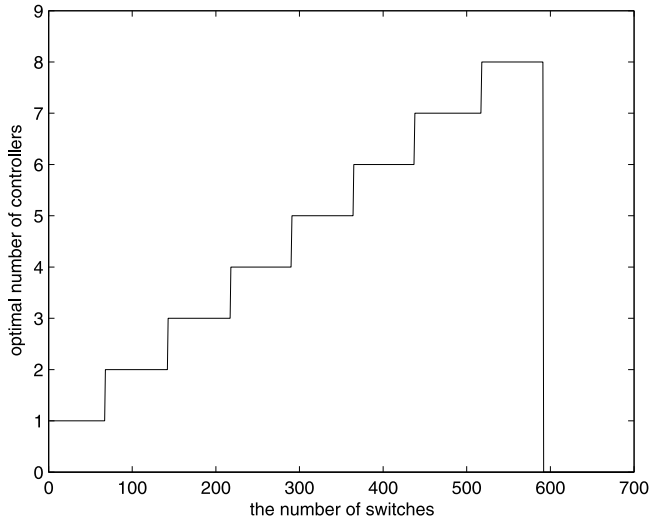


Fig. 12. The optimal number of controllers for increasing number of OpenFlow switches.

smaller increase of packet arrival rate with its increasing at each switch. In particular, it is optimal to deploy 1, 2, 3, 4, 5, 6, 7, 8 controllers respectively if each switch receives above 13453, 28032, 35680, 36043, 36096, 36109, 36112, 36113 packets per second. Besides, there is no solution if packet arrives at each switch with the rate in excess of 36113 per second. This is attributed to the fact that high packet arrival rate will lead to large packet switching delay close or even above the maximum delay allowed by quality-of-service, and it does not make sense to add more controllers.

8. Conclusion

The SDN paradigm offers numerous benefits for wide-area networks, like promoting data transfer efficiency, improving application performance and reducing deployment costs. However, it also incurs the performance bottlenecks of logically centralized controllers, and brings about an inherent penalty to essential network performance such as packet forwarding delay. Understanding the performance limitation of software-defined WAN is a prerequisite of its deployments. This paper is motivated to propose an accurate queueing system of packet forwarding to derive average packet delay through an OpenFlow switch, and build an optimization model of controller cluster deployments to solve the optimal number of controllers in software-defined WAN.

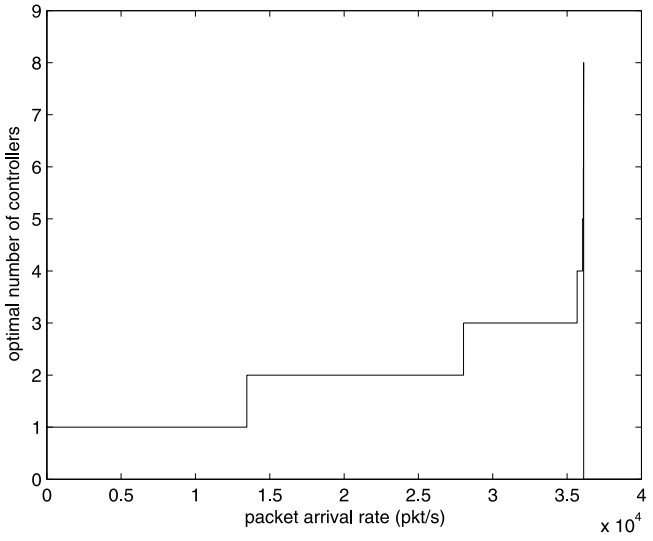


Fig. 13. The optimal number of controllers for increasing packet arrival rate at each switch.

Experimental measurements with the benchmark OFsuite_Performance indicate that our proposed queueing model offers a more precise approximation of controller cluster performance than existing ones. Furthermore, The numerical analysis on our proposed queueing model illustrates that packet forwarding performance in software-defined WAN greatly relies on packet-in message processing rate of the controller cluster, the ratio of proactively installed flow rules, the number of switches and packet arrival rate at each switch. The solution of the optimization model reveals that the optimal number of controllers is much more sharply impacted by packet arrival rate at each switch and packet-in message processing rate of a controller, than the number of switches and the ratio of proactively installed flow rules.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61572525, 61502056, 61602525), Hunan Provincial Natural Science Foundation of China (2015JJ3010), and Scientific Research Fund of Hunan Provincial Education Department, PR China (15B009, 14C0285). A preliminary version of the paper has been published in the 21st IEEE International Conference on High Performance Computing and Communications (HPCC 2019), Zhangjiajie, Hunan, China, 2019: 106-113.

References

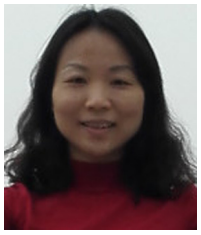
[1] N. McKeown, T. Anderson, H. Balakrishnan, et al., OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.

[2] J. Zhao, Z. Hu, B. Xiong, et al., Accelerating packet classification with counting bloom filters for virtual OpenFlow switching, China Commun. 15 (10) (2018) 117–128.
[3] O. Michel, E. Keller, SDN in wide-area networks: A survey, in: 4th International Conference on Software Defined Systems, SDS, Valencia, Spain, 2017, pp. 37–42.
[4] B. Xiong, R. Wu, J. Zhao, et al., Efficient differentiated storage architecture for large-scale flow tables in software-defined wide-area networks, IEEE Access 7 (1) (2019) 141193–141208.
[5] S. Jain, A. Kumar, S. Mandal, et al., B4: experience with a globally-deployed software defined WAN, in: 2013 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM, Hong Kong, China, 2013, pp. 3–14.
[6] C.Y. Hong, S. Mandal, A.F. Mohammad, et al., B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN, in: 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM, Budapest, Hungary, 2018, pp. 74–87.
[7] A. Tavakoli, M. Casado, T. Koponen, et al., Applying NOX to the datacenter, in: ACM Workshop on Hot Topics in Networks, HotNets, New York, USA, 2009, pp. 1–6.
[8] S. Kandula, S. Sengupta, A. Greenberg, et al., The nature of data center traffic: measurements & analysis, in: ACM SIGCOMM Conference on Internet Measurement, IMC, Chicago, Illinois, USA, 2009, pp. 202–208.
[9] L. Girish, S.K.N. Rao, Mathematical tools and methods for analysis of SDN: A comprehensive survey, in: 2nd International Conference on Contemporary Computing and Informatics, IC3I, Noida, India, 2016, pp. 718–724.
[10] Q. Zuo, M. Chen, P. Jiang, Delay evaluation of openflow control plane by queue model, J. Huazhong Univ. Sci. Technol. (Natural Science Edition) 8 (1) (2013) 44–49.
[11] L. Yao, P. Hong, W. Zhou, Evaluating the controller capacity in software defined networking, in: 23rd International Conference on Computer Communication and Networks, ICCCN, Shanghai, China, 2014, pp. 1–6.
[12] C. Metter, M. Seufert, F. Wamser, et al., Analytic model for SDN controller traffic and switch table occupancy, in: 12th International Conference on Network and Service Management, CNSM, Montreal, Canada, 2016, pp. 109–117.
[13] C. Metter, M. Seufert, F. Wamser, et al., 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) (2017) 603–615.
[14] K. Mahmood, A. Chilwan, O. Osterbo, et al., Modelling of OpenFlow-based software-defined networks: the multiple node case, IET Netw. 4 (5) (2015) 278–284.
[15] T. Li, H. Zhou, H. Luo, et al., Modeling software defined satellite networks using queueing theory, in: 2017 IEEE International Conference on Communications, ICC, Paris, France, 2017, pp. 1–6.
[16] M. Jarschel, F. Lehrieder, Z. Magyari, et al., A flexible OpenFlow-controller benchmark, in: 1st European Workshop on Software Defined Networking, EWSDN, Darmstadt, Germany, 2012, pp. 48–53.
[17] A. Shalimov, D. Zuikov, D. Zimarina, et al., Advanced study of SDN/OpenFlow controllers, in: 9th Central and Eastern European Software Engineering Conference in Russia, Moscow, Russian, 2013, pp. 1–4.
[18] Z. Shang, H. Wu, K. Wolter, An OpenFlow controller performance evaluation tool, in: 15th European Workshop on Performance Engineering, Paris, France, 2018, pp. 235–249.
[19] G. Wang, J. Li, X. Chang, Modeling and performance analysis of the multiple controllers' approach in software defined networking, in: 23rd IEEE International Symposium on Quality of Service, IWQoS, Portland, OR, USA, 2015, pp. 73–74.
[20] Y. Fu, J. Bi, J. Wu, et al., A dormant multi-controller model for software defined networking, China Commun. 11 (3) (2014) 45–55.
[21] J. Zhao, Z. Hu, B. Xiong, et al., Performance modelling and optimization of controller cluster deployments in software-defined WAN, in: 21st IEEE International Conferences on High Performance Computing and Communications, HPCC, Zhangjiajie, China, 2019, pp. 1–8.
[22] K. Sood, S. Yu, Y. Xiang, Performance analysis of software-defined network switch using M/Geo/1 model, IEEE Commun. Lett. 20 (12) (2016) 2522–2525.
[23] W. Miao, G. Min, Y. Wu, et al., Performance modelling of preemption-Based packet scheduling for data plane in software defined networks, in: IEEE International Conference on Smart City, Chengdu, China, 2015, pp. 60–65.
[24] W. Miao, G. Min, Y. Wu, et al., Performance modelling and analysis of software-defined networking under bursty multimedia traffic, ACM Trans. Multimedia Comput. Commun. Appl. 12 (5s) (2016) 77–97.
[25] Y. Ren, L. Liu, L. Cui, et al., QoS evaluation of prioritized data plane service employing queueing model, in: 25th IEEE/ACM International Workshop on Quality of Service, IWQoS, Vilanova i la Geltru, Spain, 2017, pp. 1–6.
[26] L. Liu, Y. Ren, L. Cui, et al., Performance measurement of data flow processing employing software defined architecture, Future Gener. Comput. Syst. 82 (2018) 235–243.
[27] M. Jarschel, S. Oechsner, D. Schlosser, et al., Modeling and performance evaluation of an openflow architecture, in: 23rd International Teletraffic Congress, ITC, San Francisco, USA, 2011, pp. 1–7.
[28] B. Xiong, K. Yang, J. Zhao, et al., Performance evaluation of openflow-based software-defined networks based on queueing model, Comput. Netw. 102 (2016) 172–185.
[29] A. Alghadhban, B. Shihada, Delay analysis of new-flow setup time in software defined networks, in: IEEE/IFIP Network Operations and Management Symposium, NOMS, Taipei, Taiwan, 2018, pp. 1–7.
[30] S.H. Yeganeh, Y. Ganjali, Kandoo: a framework for efficient and scalable offloading of control applications, in: ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN), Helsinki, Finland, 2012, pp. 19–24.
[31] S.H. Yeganeh, A. Tootoonchian, Y. Ganjali, On scalability of software-defined networking, IEEE Commun. Mag. 51 (2) (2013) 136–141.
[32] C. Williamson, Internet traffic measurement, IEEE Internet Comput. 5 (6) (2001) 70–74.
[33] S. Floyd, V. Paxson, Difficulties in simulating the internet, IEEE/ACM Trans. Netw. 9 (4) (2001) 392–403.
[34] C. Barakat, P. Thiran, G. Iannaccone, et al., Modeling internet backbone traffic at the flow level, IEEE Trans. Signal Process. 51 (8) (2003) 2111–2124.
[35] M.A. Arfeen, K. Pawlikowski, D. Mcnickle, et al., Towards a combined traffic modeling framework for access and core networks, in: 9th International Australasian Telecommunication Networks and Applications Conference, ATNAC, Brisbane, Queensland, Australia, 2012, pp. 1–7.
[36] M.A. Arfeen, K. Pawlikowski, A. Willig, et al., Internet traffic modelling: from superposition to scaling, IET Netw. 3 (1) (2014) 30–40.
[37] J. Cao, W.S. Cleveland, D. Lin, et al., On the nonstationarity of Internet traffic, in: 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Cambridge, Massachusetts, USA, 2001, pp. 102–112.
[38] Z.L. Zhang, V.J. Ribeiro, S. Moon, et al., Small-time scaling behaviors of Internet backbone traffic: an empirical study, in: 22nd IEEE International Conference on Computer Communications, INFOCOM, San Francisco, California, USA, vol. 3, 2003, pp. 1826–1836.
[39] V.J. Ribeiro, Z.L. Zhang, S. Moon, et al., Small-time scaling behavior of internet backbone traffic, Comput. Netw. 48 (3) (2005) 315–334.
[40] T. Karagiannis, M. Molle, M. Faloutsos, et al., A nonstationary Poisson view of Internet traffic, in: 23rd IEEE International Conference on Computer Communications, INFOCOM, Hong Kong, China, vol. 3, 2004, pp. 1558–1569.

Jinyuan Zhao, received her M.S. degree from Central China Normal University, China, in 2007. She is a PhD candidate in the School of Computer Science and Engineering of Central South University, China, and also a lecturer in the School of Computer and Communication, Hunan Institute of Engineering, China. Her current research is in software-defined networking and cloud computing.

Zhigang Hu, received his M.S. and Ph.D. degrees in Computer Science from the Central South University, China, in 1994 and 2001 respectively. He is currently a Professor in the School of Computer Science and Engineering of the Central South University. His research interests are cloud computing, grid computing, parallel and distributed systems.

Bing Xiong, received his B.S. degree in 2004 from the School of Computer Science and Technology, Hubei Normal University, China, and Ph.D. degree in 2009 by master-doctorate program from the School of Computer Science and Technology, Huazhong University of Science and Technology, China. He is currently an associate professor in the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. His main research interests include software-defined networking, network security and network measurements.



Liu Yang, received her M.S. degree in Software Engineering & Theory and Ph.D. degrees in Computer Science from Central South University, China, in 2005 and 2011 respectively. She is currently an associate professor in School of Computer Science and Engineering of Central South University. Her research interests are semantic information processing and cloud computing.

Keqin Li, a SUNY Distinguished Professor of computer science with the State University of New York, and also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 660 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.