How Powerful Switches Should be Deployed: A Precise Estimation Based on Queuing Theory

Gengbiao Shen*, Qing Li†‡, Shuo Ai*, Yong Jiang*, Mingwei Xu§, Xuya Jia*
*Graduate School at Shenzhen, Tsinghua University, Shenzhen, China
†Southern University of Science and Technology, Shenzhen, China
‡PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
§Department of Computer Science and Technology, Tsinghua University, Beijing, China
Emails: sgb16@mails.tsinghua.edu.cn, liq8@sustc.edu.cn, ais16@mails.tsinghua.edu.cn,
jiangy@sz.tsinghua.edu.cn, xmw@cernet.edu.cn, jxy14@mails.tsinghua.edu.cn

***Abstract*—Software-Defined Networking (SDN) provides a tractable and efficient architecture for operators to customize their network functions. Many traditional Data Center Networks (DCNs) are upgraded by SDN to improve link utilization and management flexibility, but they are lack of the instructions for selecting the substitutive SDN switches with the proper flow table space to achieve cost-effective and energy-saving networks. In this paper, we fill the gap of solving the flow table space estimation problem based on queuing theory. First, we divide the life process of a flow table entry into the packet-in process, the handling process and the serving process to establish a queuing system to estimate the least required number of the flow table entries of SDN switches. Second, we analyze the traffic distribution of DCNs to calculate the critical parameters in our model. Third, on the basis of the essence of the structured topologies in DCNs, we construct a probability model of routing strategies to quantize the influence of path selection. Comprehensive experiments show that the relative flow table space estimation error of our model can be less than 10%, which can give operators insights into the requirement of the SDN switches at specific positions.**

I. Introduction

With the ever-increasing requirement of improving network controllability and flexibility, Software-Defined Networking (SDN), as a promising networking paradigm, has been widely used in Wide Area Networks (WANs) [1, 2] and Data Center Networks (DCNs) [3, 4]. It separates control plane from data plane and delegates most network functions, e.g., topology discovery, traffic engineering and load balancing, to a centralized controller, while leaving only simple matching and forwarding functions at switches [5, 6]. Having a global view and fine-grain control makes the controller capable of further enhancing network performance. Considering the advantages of SDN, traditional DCNs are gradually upgraded by SDN to improve throughput and realize flexible management. However, operators have some difficulties in selecting the substitutive SDN switches with appropriate performance to execute forwarding and network functions, even though they have historical traffic statistics and the information of networks.

In SDN-based DCNs, customized SDN switches utilize the specific south-band interface, prevalently OpenFlow protocol [7], to communicate with the controller. These switches generally adopt Ternary Content Addressable Memory (TCAM) to store flow table entries for achieving high throughput and fast packet processing [8, 9]. Recent measurements show that the number of concurrent active flows in DCNs can be up to 10,000 per second, while the traffic distribution is usually inclined because of particular service deployments or asymmetric network scenarios [10, 11]. Thus, the switches at different positions require distinct TCAM demands. In practice, SDN switches have the limited capacity of TCAM and can only support several thousands of flow table entries [12], since TCAM chip is expensive (US$350 for a 1 M-bit chip) and power-hungry (15 Watt/1 Mbit) [13, 14]. Assigning all switches with equal TCAM resources (i.e., flow table space) is unnecessary and wasteful. Supplying a switch with excessive flow table space not only increases additional expense, but also causes redundant power overhead. Consequently, estimating the least required flow table space is crucial for operators to select the appropriate switches in networks for reducing capital investment and energy consumption.

To date, there are several works to construct the analytical model of the OpenFlow-based SDN or characterize the performance of routing strategies. Some researchers establish analytical models to evaluate the queuing length and the packet sojourn time in SDN [15, 16]. Others model the communication between the controller and switches as a queuing system to estimate the buffer size of the controller and the setup delay of a flow table entry [17–20]. Meanwhile, revealing the essence of routing strategies has attracted substantial attentions. Building the systematic model of routing strategies provides insights into the principles of traffic engineering [21–23]. To the best of our knowledge, there is no work to utilize queuing theory to solve the flow table space estimation problem with both SDN and routing strategies taken into consideration. Since the flow table space is proportional to the number of flow table entries with a fixed ratio, we can simply utilize the number of flow table entries to evaluate the corresponding flow table space. To solve the aforementioned problems, it is intuitive to aggregate the previous works to establish an accurate analytical model to characterize the quantity variation of flow table entries, which can realize the selection of proper performance and facilitate the identification of hotspot switches.

Corresponding author: Qing Li (liq8@sustc.edu.cn).

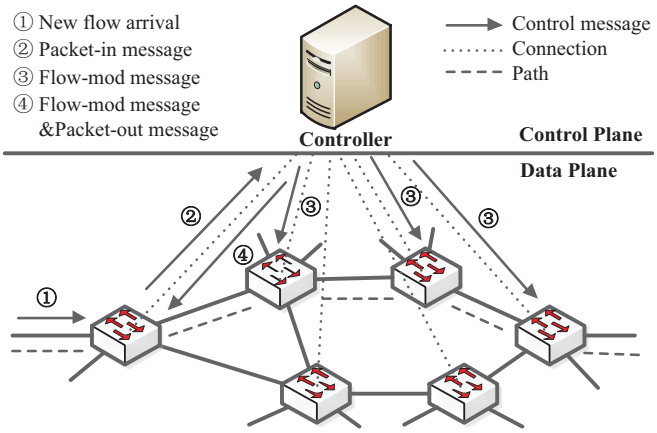


Fig. 1: The architecture of SDN

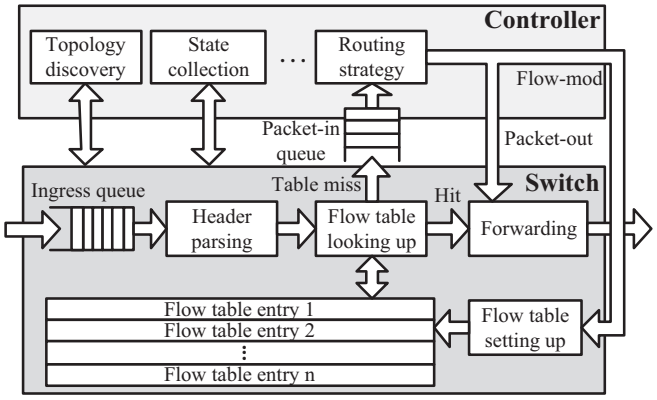


Fig. 2: The life process of a flow table entry

In this paper, we aim to fill the gap of estimating the least required flow table space of the SDN switches in DCNs. First, we divide the whole process from new flows arriving at the switch to the flow table entry expiring into three parts: the packet-in process that new flows trigger packet-in messages, the handling process that the controller makes routing decisions, and the serving process that the flow table entry serves for forwarding. We model these three parts as $M/M/1/\infty$, $M/M/1/\infty$, and $M/G/c/c$ systems respectively, and utilize the synthetical model to derive the least required number of flow table entries under the constraint of the upper bound probability of failing to establish the path. Then we make some practical traffic distribution assumptions and calculate the critical parameters in our model through analyzing the real workloads in DCNs. Last, we take full advantages of the structured topologies in DCNs to establish a probability model of routing strategies to quantize the influence of path selection. We conduct comprehensive experiments to evaluate the performance of our model under various model parameters and different workloads. The results show that our model can estimate the least required flow table space of different switches at specific positions and achieve little relative flow table space estimation error that can be less than 10%, which can give operators insights into the performance requirement of the SDN switches in DCNs.

In summary, our contributions are three-fold:

- We are the first to utilize queuing theory to establish a queuing system to estimate the least required flow table space of the SDN switches in SDN-based DCNs.
- We make some practical assumptions through observing the real workloads in DCNs and calculate the critical parameters in our model.
- We set up a probability model of routing strategies based on the essence of the structured topologies in DCNs.

The rest of the paper is organized as follows. Section II describes the analytical models of queuing systems in SDN. In Section III, we calculate the corresponding parameters in our estimation model under the assumptions of the traffic distribution and the model of routing strategies in DCNs. In Section IV, we conduct simulation and discuss the performance of our estimation model. Section V states the conclusion.

II. Analytical Model

A typical architecture of the OpenFlow-based SDN includes one centralized controller and some customized switches as shown in Fig. 1. SDN separates the network into a control plane that contains a centralized controller with various applications for global control and a data plane that forwards packets and collects network states. The switches connect to the controller directly through dedicated links. When a packet arrives at a switch, the switch looks up its flow table to find the matching rule. If the switch finds the result, it performs the corresponding actions, otherwise it encapsulates the packet into a packet-in message sent to the controller. Then the controller generates routing path and forwarding actions and returns control messages including the flow-mod message used to set up a flow table entry and the packet-out message used to put the packet back to the network. A flow table entry has the limited lifetime and it expires when the condition of the idle timeout or the hard timeout is satisfied [24].

As illustrated above, the whole life process of a flow table entry can be abstracted into three parts:

- **The packet-in process**. This process is from the new flow packet arriving at a switch to the switch sending the corresponding packet-in message.
- **The handling process**. This process is from the controller receiving packet-in messages to the controller making decisions and sending control messages to the corresponding switches.
- **The serving process**. This process is from the switch receiving control messages and setting up a flow table entry to the flow table entry expiring.

Combining with the structures of the OpenFlow-based SDN switch and the controller, the life process of a flow table entry is depicted in Fig. 2. Then we model the three parts as queuing systems to calculate the least required number of flow table entries of each switch.

A. Model of the Packet-in Process

For the OpenFlow-based SDN, the switch typically has a single processor which provides enough processing capacities to execute looking up, matching and forwarding. In the network, an edge switch receives new flow packets from its multiple ports that connect to several subnets, and aggregates them into an ingress queue to wait for the processor idle time. Since the switch executes the header parsing and the flow table looking up independently, the processing rate is stable generally. Thus, for an edge switch $v_e \in V_e$, where V_e is the set of edge switches in the network, we suppose the processing time of v_e generating the packet-in message for a new flow packet as a random variable that conforms to the negative exponential distribution with parameter $\mu^{(v_e)}$.

Previous works suppose that the packet arrival in switches conforms to the Poisson process [16, 20] or the compound Poisson process [19]. In practice, different applications have distinct traffic features. For example, remote login and file transfer can be well modeled as the Poisson process, but others conform to the compound Poisson process because of the traffic burst [25]. Moreover, clustered application hosts in DCNs generate the relatively stable and periodic traffic, which simplifies modeling the traffic. In our model, we only focus on the first packets of new flows that trigger the table miss and the packet-in message generating in the switch. Hence, for an edge switch v_e with $n_{ep}^{(v_e)}$ external ports, i.e., the ports that connect to external networks, we suppose the arrivals of new flows from different external ports are independent and conform to the Poisson process with rate $\lambda_i^{(v_e)}(i = 1, \cdots, n_{ep}^{(v_e)})$ respectively. Here, we consider all packets that trigger the switch generating packet-in messages as the arrivals of new flows regardless of whether they result from the actual new flow arriving or the flow table entry expiring. According to [26], a process that consists of multiple Poisson processes is also a Poisson process, and the rate of the process is the sum of the rates of all Poisson processes. So the synthetical arrival rate of new flows in v_e is $\lambda_{in}^{(v_e)} = \sum_{i=1}^{|n_{ep}^{(v_e)}|} \lambda_i^{(v_e)}$.

After new flows arrivie at an edge switch, the packets of flows are cached in the switch to wait for the setup of the flow table entry. As the performance of the switch improves, the switch has sufficient storage space and enough processing capacities to handle this situation. Thus, for simplicity, we can assume that during the waiting process the switch can cache all packets of new flows without limitation. Combining with the previous discussion, we can model the packet-in process as a $M/M/1/\infty$ system. According to [26], the departure process of a $M/M/1/\infty$ system is also a Poisson process with the rate identical to the rate of the arrival process. Hence, the departure process of the packet-in messages generated by v_e conforms to the Poisson process with rate $\lambda_{out}^{(v_e)} = \lambda_{in}^{(v_e)}$.

B. Model of the Handling Process

The centralized controller in SDN manages all switches in the network. It aggregates the packet-in messages received from edge switches into a packet-in queue, and handles the messages in order independently. The controller makes routing decisions according to the routing strategy and the network states obtained from the modules of topology discovery and state collection. Then the controller sends flow-mod messages to the corresponding switches along the forwarding path. The model of this process is described in Fig. 3. Generally, the centralized controller is a high performance server that can handle packet-in messages with steady rate. Thus, we suppose the duration of the controller handling a packet-in message conforms to the negative exponential distribution with parameter $\mu^{(c)}$. Meanwhile, the arrival rate of the packet-in messages from v_e can be defined as $\lambda_{pi}^{(v_e)}$ which is equal to the corresponding departure rate, i.e, $\lambda_{pi}^{(v_e)} = \lambda_{out}^{(v_e)}$. Based on the previous discussion, the synthetical arrival rate of the packet-in messages in the controller can be expressed as

$$\lambda_{pi}^{(c)} = \sum_{v_e=1}^{|V_e|} \lambda_{pi}^{(v_e)} = \sum_{v_e=1}^{|V_e|} \lambda_{out}^{(v_e)} = \sum_{v_e=1}^{|V_e|} \lambda_{in}^{(v_e)}$$

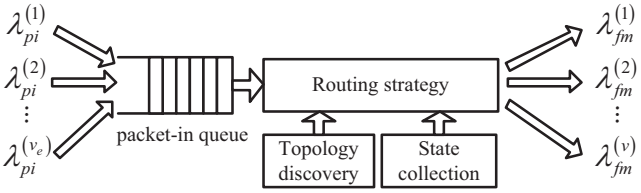


Fig. 3: The queuing model of the handing process

In addition, the cache capacity in the controller can be regarded as infinity, since the size of packet-in messages is not large (normally 128 bytes [24]) and the number of concurrent new flows is no more than tens of thousands [10]. Hence, the handling process can be modeled as a $M/M/1/\infty$ system. The controller handles packet-in messages to find the most appropriate paths for flows and generates multiple flow-mod messages for the corresponding switches. Similar to the previous discussion, the departure rate of flow-mod messages $\lambda_{fm}^{(c)}$ is identical to the arrival rate of the synthetical packet-in messages $\lambda_{pi}^{(c)}$, i.e., $\lambda_{fm}^{(c)} = \lambda_{pi}^{(c)}$. In the handling process, the departure process of flow-mod messages is a batch departure process, because the routing strategy generates the routes that contain multiple switches. The batch size depends on the selected path and the width of the network.

For a switch $v \in V$, where V is the set of switches, we suppose its arrival rate of flow-mod messages is $\lambda_{fm}^{(v)}$. When the network deploys a known routing strategy, we define the probability of the flow-mod messages that originate from the edge switch v_e and are to be sent to the switch v as $\pi_v^{(v_e)}$. Hence, combining with the model of the packet-in process, the arrival rate of flow-mod messages in v can be expressed as $\lambda_{fm}^{(v)} = \sum_{v_e=1}^{|V_e|} \pi_v^{(v_e)} \lambda_{pi}^{(v_e)}$, where $\pi_v^{(v_e)} \in (0, 1)$. The arrival of flow-mod messages conforms to the Poisson process.

C. Model of the Serving Process

Once receiving a flow-mod message, the SDN switch either establishes the corresponding flow table entry or returns an

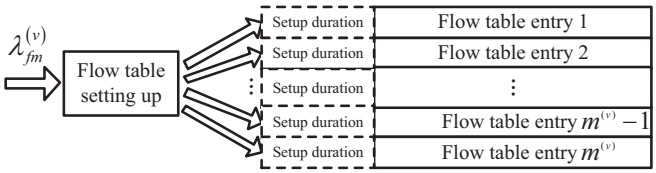


Fig. 4: The queuing model of the serving process

error message to the controller while there is no enough flow table space. So the serving process can be divided into two parts. One is the process of the switch establishing the flow table entry. Another is the process of the flow table entry expiring. For the first part, the switch sets up flow table entries in order of the arrivals of flow-mod messages. All setup durations are independent and related to the number of the existing flow table entries and the inserted positions of new flow table entries[12]. In steady states, the duration of setting up a flow table entry is relatively stable, since the number of flow table entries and the inserted position of the new flow table entry have no significant changes. For simplicity, we ignore the rule aggregation in the switch. And we assume all setup durations of flow table entries are identical and constant and suppose the average duration of this part as $\bar{t}_s^{(v)}$ for v.

The second part is the timeout process of the flow table entry. Each flow table entry contains an idle timeout and a hard timeout [24]. The flow table entry will be removed when it exists for the hard timeout duration or the time interval of matching it exceeds the idle timeout duration. Generally, all flow table entries in the network have the identical idle timeout and hard timeout. We suppose the idle timeout and the hard timeout are o_{idle}, o_{hard} respectively. Hence, the average duration of this part depends on the traffic distribution and the timeout values. We suppose the average duration of this part as $\bar{t}_d^{(v)}$ for the switch v.

Thus, the average life duration of a flow table entry can be expressed as $\bar{t}_f^{(v)} = \bar{t}_s^{(v)} + \bar{t}_d^{(v)}$, which indicates that the average serving rate of a flow table entry is $\mu_f^{(v)} = 1/\bar{t}_f^{(v)}$. Obviously, this serving rate is a random variable which conforms to the general distribution. The previous discussion shows that the arrival of flow-mod messages conforms to the Poisson process with rate $\lambda_{fm}^{(v)}$. Moreover, the arrival of a flow-mod message occupies one flow table entry in advance. And the switch can only hold the limited number of flow table entries related to the flow table space without caching extra flow-mod messages. Hence, we can model the serving process as a $M/G/c/c$ system, as shown in Fig. 4. We suppose the maximum number of flow table entries that can be stored in the flow table of the switch v as $m^{(v)}$, which implies that different switches have distinct serving capacities with parameter $c^{(v)} = m^{(v)}$. For a $M/G/c^{(v)}/c^{(v)}$ system, we use $p_n^{(v)}$ to denote the steady-state probability of having n flow table entries in the switch v. According to [26], this probability can be expressed as

$$p_n^{(v)} = \frac{(\lambda_{fm}^{(v)}/\mu_f^{(v)})^n/n!}{\sum_{i=0}^{c^{(v)}}(\lambda_{fm}^{(v)}/\mu_f^{(v)})^i/i!}$$

where $n \in \{0, 1, \cdots, c^{(v)}\}$.

Combining with the previous models, the flow table setup failure probability which denotes the probability of failing to respond the flow-mod action can be expressed as

$$p_{loss}^{(v)} = p_{c^{(v)}}^{(v)} = p_{m^{(v)}}^{(v)} = \frac{(\lambda_{fm}^{(v)}/\mu_f^{(v)})^{m^{(v)}}/(m^{(v)})!}{\sum_{i=0}^{m^{(v)}}(\lambda_{fm}^{(v)}/\mu_f^{(v)})^i/i!}$$

where $\lambda_{fm}^{(v)} = \sum_{v_e=1}^{|V_e|} \pi_v^{(v_e)} \sum_{j=1}^{|n_{ep}^{(v_e)}|} \lambda_j^{(v_e)}$.

This failure probability decreases as the serving capacity increases. For each switch v, the maximum flow table setup failure probability can be defined as $\eta^{(v)}$. We define the Upper Bound Probability of Failing to establish the path (UBPF) as ξ. For a network G with the structured topology, we can obtain $\eta^{(v)} = \Gamma(G, v, \xi)$, where $\Gamma(\cdot)$ is related to the topology and the routing strategy. In order to satisfy the UBPF, the flow table setup failure probability of every switch cannot exceed the corresponding maximum flow table setup failure probability. Consequently, for the network G and the given UBPF ξ, we can solve (1) to obtain the least required number of flow table entries $m_{least}^{(v)}$ of the switch v.

$$m_{least}^{(v)} = \min_{m^{(v)}}(p_{loss}^{(v)} \leq \Gamma(G, v, \xi)) \quad (1)$$

Then we suppose the storage size of a flow table entry is B in the given OpenFlow version. Since the flow table space is proportional to the number of flow table entries, we can obtain that the least required flow table space of the switch v is $T^{(v)} = m_{least}^{(v)} B$ which can be used to evaluate the least required flow table space of the switches at different positions. Since the storage size of a flow table entry is fixed and determined by the OpenFlow version that the network adopts, we can briefly utilize (1) to evaluate the least required flow table space. The previous discussion indicates that the probability $\pi_v^{(v_e)}$ depends on which routing strategy that the network deploys and the average serving rate $\mu_f^{(v)}$ is related to the traffic distribution of the network. Thus, we make some assumptions and model the routing strategies to calculate the corresponding parameters in the next section.

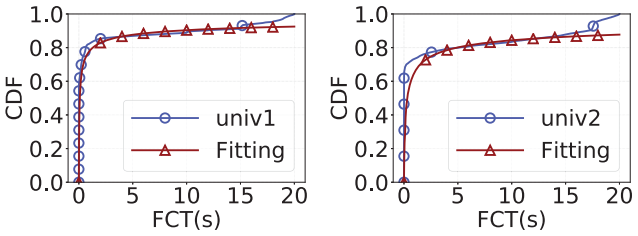
III. Parameters Calculating

A. Traffic Distribution Assumption

1) Flow Completion Time in DCNs: Traffic distribution in DCNs is relatively regular, stable and predictable [27]. The crucial flow characteristics consist of Flow Completion Time (FCT) and the ingress-egress pairs. The previous work shows that the traffic size in DCNs conforms to the distribution with heavy tail [10], which means that most flows are the short flows with small FCTs but others are the long flows with huge FCTs. Hence, the FCT distribution of flows can be modeled as the Pareto Type II distribution that can be expressed as

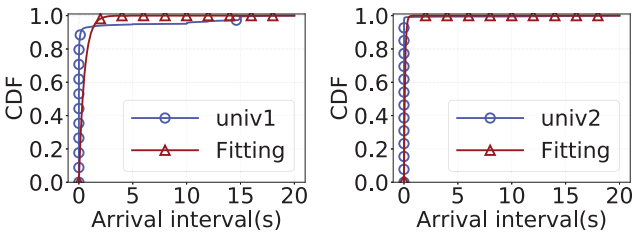
$$F(t) = 1 - [1 + \frac{t - t_{min}}{\sigma}]^{-\alpha}, (t >= t_{min})$$

where $F(t)$ is the cumulative distribution function of the flows with the FCT of t s, t_{min} is the minimum FCT of the flows



(a) univ1 (b) univ2

Fig. 5: FCT distribution



(a) univ1 (b) univ2

Fig. 6: Packet arrival interval of the type III flow

that can be set as the round trip time t_{rtt} of the network, $\sigma > 0$ is the scale factor, and $\alpha > 1$ is the shape factor.

To validate our FCT assumption, we analyze the traffic distribution of two real workloads, i.e., univ1 and univ2 in [10]. The fitting results are shown in Fig. 5. We assume the idle timeout and the hard timeout are $o_{idle} = 10s$ and $o_{hard} = 20s$ respectively. In our analysis, we regard the arrival of packets after the hard timeout as the arrival of new flows, so the flow that lasts for more than o_{hard} is split into several short flows. Fig. 5a shows that the FCT distribution in univ1 matches with the Pareto Type II distribution with parameters $\sigma = 0.015, \alpha = 0.36, t_{min} = 3.815 \times 10^{-6}s$. Fig. 5b shows that the FCT distribution in univ2 matches with the Pareto Type II distribution with parameters $\sigma = 0.05, \alpha = 0.35, t_{min} = 9.54 \times 10^{-7}s$. Obviously, the fitting results demonstrate that our FCT assumption is reasonable.

2) Average Serving Rate of the Flow Table Entry: The transfer duration of a flow and the inter-arrival behavior of the packets in a flow depend on the different type of applications that the flow belongs to. In most scenarios, the durations of short flows are smaller than the idle timeout of the flow table entry, while the durations of long flows are larger. Hence, all flows can be divided into three types:

- **Type I: Short flow.** The FCT of this type flow is smaller than the idle timeout.
- **Type II: Long flow with small interval.** The FCT of this type flow is larger than the idle timeout, and the maximum packet arrival interval in this type flow is smaller than the idle timeout.
- **Type III: Long flow with large interval.** The FCT of this type flow is larger than the idle timeout, but the maximum packet arrival interval in this type flow is larger than the idle timeout.

For the type I flow, the expected timeout duration of the flow table entry can be express as

$$\begin{aligned} E_1[t_d^{(v)}] &= min(\int_{t_{min}}^{o_{idle}} tF'(t) + o_{idle}, o_{hard}) \\ &= min(\frac{\sigma}{\alpha - 1} + t_{min} - \varepsilon + o_{idle}, o_{hard}) \end{aligned} \tag{2}$$

where

$$\varepsilon = (o_{idle} + \frac{\sigma + o_{idle} - t_{min}}{\alpha - 1})(1 + \frac{o_{idle} - t_{min}}{\sigma})^{-\alpha}$$

Then, we suppose the proportion of long flows that belong to the type II is φ. Since the packet arrivals of the type II flow do not trigger the idle timeout, the expected timeout duration of the flow table entry can be expressed as

$$E_2[t_d^{(v)}] = [1 - F(o_{idle})]\varphi o_{hard} \tag{3}$$

The type III flow is the flow that lasts for a long time and has the remarkable on-off characteristic. The applications that this type flows belong to are mostly remote login and file transfer. Hence, we can model the type III flow as the Poisson stream with rate γ_p, which indicates that the cumulative distribution of the packet arrival interval of this type flow is $A(\Delta t) = 1 - e^{-\gamma_p \Delta t}$, where Δt is the packet arrival interval.

We analyze the packet arrival interval distribution of the type III flow in the two real workloads. Fig. 6a shows that the packet arrival interval distribution of the type III flow in univ1 matches with the negative exponential distribution with parameter $\gamma_p = 2$. Fig. 6b shows that the packet arrival interval distribution of the type III flow in univ2 matches with the negative exponential distribution with parameter $\gamma_p = 5$. The fitting results demonstrate that our assumptions conform to real scenarios.

The timeout duration of the flow table entry depends on the first time when the arrival interval is larger than the idle timeout. We assume that there are $k - 1$ arrivals before it happens. The probability that the interval between the kth and $(k-1)th$ arrival is larger than the idle timeout conforms to the geometric distribution with parameter $p = e^{-\gamma_p o_{idle}}$. It is easy to obtain that the expected arrival number is $1/p = e^{\gamma_p o_{idle}}$. So the expected life duration of the flow table entry before the first interval larger than the idle timeout is

$$E_r = e^{\gamma_p o_{idle}} \int_{\Delta t=0}^{o_{idle}} tA'(\Delta t) = \frac{1}{\gamma_p}(e^{\gamma_p o_{idle}} - 1) - o_{idle}$$

Hence, the expected timeout duration of the flow table entry of the type III flow can be expressed as

$$E_3[t_d^{(v)}] = [1 - F(o_{idle})](1 - \varphi) \times min(E_r + o_{idle}, o_{hard}) \tag{4}$$

All flow table entries in switches serve for the flows belong to the three types. Therefore, the average duration of a flow table entry can be expressed as $\bar{t}_d^{(v)} = \sum_{l=1}^{3} E_l[t_d^{(v)}]$. Hence, the average serving rate of the flow table entry is

$$\mu_f^{(v)} = \frac{1}{\bar{t}_f^{(v)}} = \frac{1}{\bar{t}_s^{(v)} + \bar{t}_d^{(v)}} = \frac{1}{\bar{t}_s^{(v)} + \sum_{l=1}^{3} E_l[t_d^{(v)}]} \tag{5}$$

3) The Trace of Flows: Our estimation model is influenced by the trace of flows. In DCNs, flows are injected from one edge switch then transferred to the other edge switch. The pairs of edge switches consist of all ingress-egress pairs, which can be used to classify the flows from the source-destination perspective. For a source edge switch $v_s \in V_e$, we define the fraction of the new flows that are injected into this switch and transferred to the destination edge switch $v_t \in V_e$ as $\omega^{(s,t)} \in [0,1]$. Hence, for every edge switch v_e, there are several parameters $\omega^{(e,t)}$ to assign the corresponding fractions of flows to all other edge switches v_t, where $\sum_{v_t \in V_e} \omega^{(e,t)} = 1$. Meanwhile, the forwarding paths of ingress-egress pairs depend on the practical routing strategy.

B. Model of Routing Strategies

Typically, DCNs have structured topologies, which can simplify modeling routing strategies. Existing models of routing strategies consider more about the dynamic of the schemes, but they are lack of the probability essence in the steady states. So they are not suitable for our estimation model. We design our model of routing strategies based on probability theory. We only consider the routing strategies with the stable path selection probability in steady states, which can characterize the probability essence of routing strategies directly. For the edge switches $v_s, v_t \in V_e$, we define the set of paths from v_s to v_t as $R^{(s,t)} = \{r^{(s,t)}(i)\}$ and the corresponding path selection probability as $\rho^{(s,t)}(i) \in [0,1]$ where $\sum_i \rho^{(s,t)}(i) = 1$. Therefore, according to the previous discussion, the probability that the switch v is selected by the new flow arriving at the edge switch v_e can be expressed as

$$\pi_v^{(v_e)} = \sum_{\substack{v_t \neq v_e \\ v_t \in V_e}} \sum_{\substack{v \in r^{(e,t)}(i) \\ r^{(e,t)}(i) \in R^{(e,t)}}} \omega^{(e,t)} \rho^{(e,t)}(i) \tag{6}$$

In our model, we utilize the widely adopted Clos topology, i.e., fat-tree [28], as an example to analyze the model of routing strategies. Other topologies can use the same methods to construct the model. In SDN-based DCNs, the controller selects one of the available equal cost paths for a new flow according to the routing strategy. For the fat-tree topology, a path can be divided into the upstream route that is from the source edge switch to the top level switch and the downstream route that is from the top level switch to the destination edge switch, while the two routes do not include the top level switch. The whole path depends on the selected upstream route, since the downstream route is determined by the top level switch. We suppose that the number of the upstream egress ports in each switch v is $n_{up}^{(v)}$. Different routing strategies have distinct selection probabilities because of various scenarios. In our model, we utilize two stable routing strategies, i.e., ECMP and WCMP that are widely adopted in DCNs, as an example to construct the model. Other routing strategies can also be modeled by this method through analyzing steady network states and considering real traffic scenarios.

ECMP utilizes the static hash mechanism to hash five-tuple packet header to select the egress port for each flow. In SDN, this mechanism is realized in the centralized controller without maintaining per-flow states. For stochastic flows, the hashing results are random, which causes the probabilities of selecting every feasible forwarding egress port for a switch on the upstream route are equal. Therefore, the possibilities of selecting each feasible path $r^{(s,t)}(i)$ are identical. For every $R^{(s,t)}$, we suppose the switch that belongs to the upstream route of the path i is $v_h \in r_{up}^{(s,t)}(i)$, where $r_{up}^{(s,t)}(i)$ is the set of all the switches on the upstream route of the path $r^{(s,t)}(i)$. Hence, the path selection probability can be expressed as $\rho^{(s,t)}(i) = \prod_{v_h \in r_{up}^{(s,t)}(i)} \frac{1}{n_{up}^{(v_h)}}$. Combining with (6), we can get the corresponding probability.

WCMP is similar to ECMP but chooses the upstream egress ports according to the prescribed weights instead of randomness. WCMP is typically utilized to surmount the asymmetric scenarios, e.g., asymmetric topology or skew traffic distribution, for balancing all flows in the network. The centralized controller can assign the proper weights for the upstream egress ports of each switch on the basis of network states and the known traffic distribution. For a switch $v \in V$ with $n_{up}^{(v)}$ upstream egress ports, we suppose the weight that the controller assigns for each upstream egress port j is $\beta_j^{(v)} \in [0,1]$, where $\sum_{j=1}^{n_{up}^{(v)}} \beta_j^{(v)} = 1$. The set of the upstream egress ports of the switch v can be defined as $N_{up}^{(v)}$. Hence, the path selection probability can be expressed as $\rho^{(s,t)}(i) = \prod_{v_h \in r_{up}^{(s,t)}(i), j \in N_{up}^{(v_h)}} \beta_j^{(v_h)}$. According to (6), we can calculate the corresponding selection probability.

Consequently, combining with (1), (2), (3), (4), (5), (6), we can obtain the least required number of flow table entries of every switch to estimate the least required flow table space of the switches at different positions.

IV. Evaluation

In this section, we evaluate our flow table space estimation model under various parameters and different workloads. We implement a simulator to imitate the behaviors of the SDN switches and the controller to collect the number of the flow table entries of all switches.

A. Experiment Setup

1) Topology: We evaluate our estimation model in the fat-tree topology that is widely adopted in DCNs. The fat-tree topology consists of multiple layer switches and the number of equal cost paths between the ingress-egress pairs is identical. In our model, we utilize fat-tree topology with parameter $k = 4$, as shown in Fig. 7.

2) Maximum Flow Table Setup Failure Probability: According to the previous discussion, for each switch v, the maximum flow table setup failure probability $\eta^{(v)}$ is related to the network G and the UBPF ξ. In the fat-tree topology, the switches can be divided into three types: the ToR switch, the aggregation switch and the core switch, on the basis of the switch positions. For simplicity, we suppose that the switches belong to the same type have the identical maximum flow table setup failure probability which can be assumed as η^e, η^a, η^c

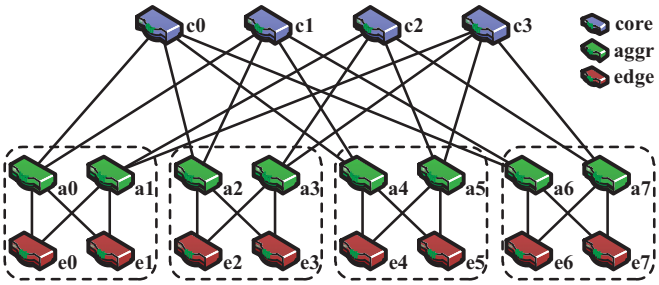


Fig. 7: $k = 4$ fat-tree topology

respectively. Furthermore, there are only three types, i.e., 1, 3 or 5, of path length in our fat-tree topology. We define the fractions of the three type paths as θ_1, θ_3, θ_5 respectively. Consequently, to satisfy the UBPF ξ, the following inequality should be satisfied

$$\theta_1[1-(1-\eta^e)] + \theta_3[1-(1-\eta^e)^2(1-\eta^a)] + \theta_5[1-(1-\eta^e)^2(1-\eta^a)^2(1-\eta^c)] \leq \xi \tag{7}$$

Combining with the previous discussion, the value of $\Gamma(G, v, \xi)$ is the acceptable solution of (7). Once the UBPF ξ is given, the $\eta^{(v)}$ of each switch v can be obtained.

3) Metrics: To evaluate the performance of our estimation model, we utilize the relative flow table space estimation error ζ between our model values and the simulation results to evaluate the precision of our model. For each switch v, we define the average top 20% number of flow table entries that is measured in the simulation as $\bar{m}^{(v)}_{least}$. Consequently, the value of ζ can be expressed as

$$\zeta = \left|\frac{\sum_{v\in V} m^{(v)}_{least} - \sum_{v\in V} \bar{m}^{(v)}_{least}}{\sum_{v\in V} \bar{m}^{(v)}_{least}}\right| \tag{8}$$

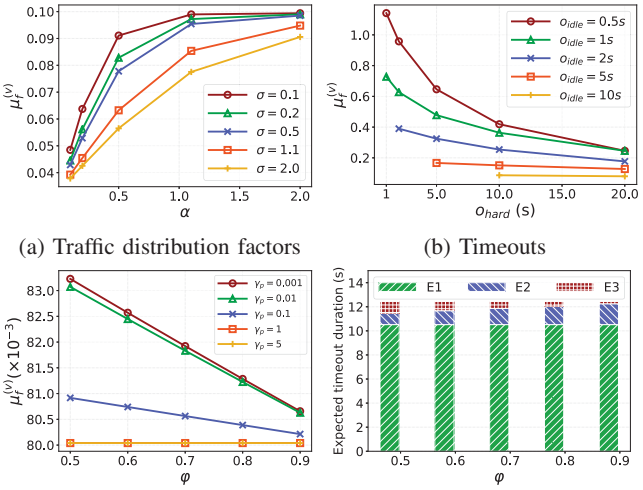
Through ζ, we can figure out the gap between the real size of flow table space and the theoretical values of our model.

4) Default Parameters: We give some default model parameters for the simulation when there are no specific instructions. We set the average setup duration of a flow table entry is $\bar{t}^{(v)}_s = 0.05s$, since the OpenFlow-based switch can establish 200 flow table entries per second on average [12]. The idle timeout and the hard timeout are set as $o_{idle} = 10s$ and $o_{hard} = 20s$ respectively. According to the previous discussion, we set the default parameters of the FCT distribution are $\alpha = 0.36$, $\sigma = 0.015$ and $t_{min} = 5 \times 10^{-6}s$. The probability that a long flow belongs to the type II flow is $\varphi = 0.6$. And the average packet arrival rate of the type III flow is $\gamma_p = 2$. We set the default flow arrival rate of each switch as $\lambda^{(v_e)}_{in} = 100$.

B. Average Serving Rate of the Switch

We first investigate the effects of different model parameters on the average serving rate of the switches in the network.

We vary the shape factor from $\alpha = 0.1$ to 2 while the scale factor is set as $\sigma = 0.1, 0.2, 0.5, 1.1, 2.0$ respectively. As shown in Fig. 8a, the average serving rate of the switches increases as the shape factor increases and the scale factor



(a) Traffic distribution factors (b) Timeouts

(c) Probability of the Type II flow (d) Expected timeout durations

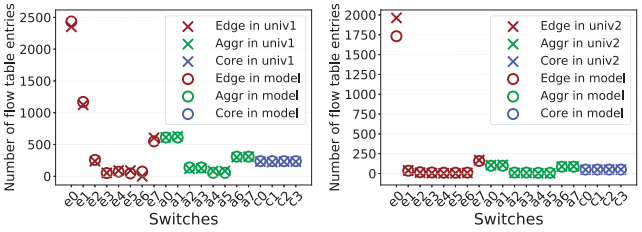
Fig. 8: Effects of different parameters on average serving rate

decreases, since the larger shape factor or the smaller scale factor causes the more intensive flows that have the smaller FCTs. Then we vary the values of the hard timeout and the idle timeout. From Fig. 8b we can see that the average serving rate decreases when the hard timeout or the idle timeout increases, because the larger timeout values result in the longer durations of flow table entries. Therefore, the hard timeout and the idle timeout of flow table entries should be selected properly in different scenarios, since the small timeout values cause the frequent flow table establishing which degrades the forwarding performance, while the large timeout values damage the serving rate of the switches. Moreover, we vary the probability of the long flows that belong to the type II from $\varphi = 0.5$ to 0.9, while the average arrival rate of the type III flow is set as $\gamma_p = 0.0001, 0.01, 0.1, 1, 5$. As shown in Fig. 8c, we can obtain that the average serving rate decreases as φ increases or γ_p increases, since it causes the more fraction of the type II flows or the longer durations of the type III flows. The expected timeout durations of the three type flows are shown in Fig. 8d, We can obtain that the expected duration of short flows is approximately 5 times longer than the expected duration of long flows, and the expected duration of the type III flow decreases as φ increases, since most of flows are the short flows that contribute most to the consumption of flow table entries in the network.

C. Real Workloads

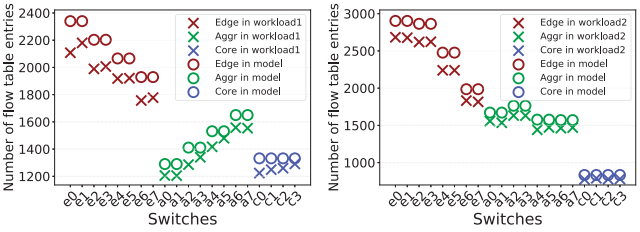
We utilize two real workload `univ1` and `univ2` both in [10] to evaluate our model. We assume the network deploys ECMP to make route decisions for each new flow, while we can utilize the same simulation method to evaluate our model in the WCMP scenarios.

1) `univ1`: We analyze `univ1` to obtain the necessary parameters in our model. According to the previous discussion about `univ1`, we can obtain the parameters of the FCT distribution are $\alpha = 0.36$, $\sigma = 0.015$ and $t_{min} = 3.815 \times 10^{-6}s$.



(a) univ1 (b) univ2

Fig. 9: Real workloads



(a) workload I (b) workload II

Fig. 10: Simulated workloads

TABLE I: The flow arrival rates of real workloads

	Flow Arrival Rates (/s)							
	e0	e1	e2	e3	e4	e5	e6	e7
univ1	156.461	58.312	8.350	2.243	2.046	1.992	1.931	16.246
univ2	113.098	0.558	0.291	0.209	0.104	0.097	0.095	0.531

TABLE II: Simulated workloads parameters

	Pods	Pod1	Pod2	Pod3	Pod4
workload I	Rack	64.0%	48.0%	32.0%	16.0%
	In-pod	16.0%	12.0%	8.0%	4.0%
	Out-pod	20.0%	40.0%	60.0%	80.0%
workload II	Rack	13.3%	2.7%	12.1%	0.0%
	In-pod	80.9%	81.3%	56.3%	30.7%
	Out-pod	5.8%	16.0%	31.6%	69.3%

And the average packet arrival rate of the type III flow is $\gamma_p = 2$. The probability that a long flow belongs to the type II flow is $\varphi = 0.6$. We assign the flows to each edge switch based on the different subnets. The flow arrival rates of all edge switches are shown in Table I, and the fractions of all ingress-egress pairs can also be obtained from univ1. We set the UBPF in univ1 as $\xi = 0.0001$. According to (7), we can get the acceptable maximum flow table setup failure probabilities of each type switch, and the corresponding probabilities are $\eta^e = 0.0001$, $\eta^a = 0.0004$ and $\eta^c = 0.0008$ respectively

We inject the traffic of univ1 into our simulator and collect the number of flow table entries of all switches every $500ms$. Then we run our model to obtain the theoretical number of flow table entries. The result is shown in Fig. 9a. We find that the theoretical result is extremely close to the real value. According to (8), we can obtain the relative flow table space estimation error $\zeta = 1.37\%$, which means that our model can precisely estimate the real requirement of the flow table space of the switches at different positions.

2) univ2: We further analyze univ2 to validate our model. The same as the previous experiment, we firstly give some critical model parameters. According to the previous discussion about univ2, we can obtain the parameters of the FCT distribution are $\sigma = 0.05$, $\alpha = 0.35$ and $t_{min} = 9.54 \times 10^{-7}s$. And the average packet arrival rate of the type III flow is $\gamma_p = 5$. The probability that a long flow belongs to the type II flow is $\varphi = 0.61$. Through analyzing univ2, we can obtain the flow arrival rates of all switches, as shown in Table I. Given the UBPF $\xi = 0.0001$, the acceptable maximum flow table setup failure probabilities of each type switch are $\eta^e = 0.0001$, $\eta^a = 0.00039$ and $\eta^c = 0.0008$ respectively.

We inject the traffic of univ2 into our simulator and collect the number of flow table entries of all switches every $500ms$. We compare the simulated results with our model results, as shown in Fig. 9b. We find that the theoretical result is still close to the real value. According to (8), we can obtain the relative flow table space estimation error $\zeta = 7.64\%$. Although the traffic in univ2 is more skewed than the traffic in univ1 which results in the larger estimation error, our model can still obtain the great estimation results.

D. Simulated Workloads

For further validating the efficiency of our model, we utilize real workload parameters to generate the simulated workloads to conduct simulations.

1) workload I: This workload parameters are from [10], as shown in Table II. We generate the traffic trace according the given workload I parameters. The fractions of all ingress-egress pairs can be obtain according to the ratio of the different type of flows. We set the rack ratio as the fraction of the same switch pair, the in-pod ratio as the fraction of the pair between the original edge switch and the other edge switch in the same pod, and the out-pod ratio as the fraction of the pair between the original edge switch and other edge switches in other pods where all other edge switches share the fraction equally. We set the UBPF in workload I as $\xi = 0.0001$, and get the corresponding probabilities $\eta^e = 0.00011$, $\eta^a = 0.00038$ and $\eta^c = 0.0008$.

We inject the traffic of workload I into our simulator and collect the number of flow table entries of all switches every $500ms$. The results of the number of flow table entries are shown in Fig. 10a. According to (8), we can obtain the relative flow table space estimation error $\zeta = 7.69\%$, which means that in the heavy-load scenario, our model can also get a good estimation for each switch.

2) workload II: This workload parameters are from [27], as shown in Table II. The same as the previous experiment, we generate the corresponding traffic trace, and the fractions of all ingress-egress pairs can be obtained according to the ratio parameters. We set the UBPF in workload II as $\xi = 0.0001$, and get the corresponding probabilities $\eta^e = 0.00009$, $\eta^a = 0.00039$ and $\eta^c = 0.0008$.

We inject the traffic of workload II into our simulator and collect the number of flow table entries of all switches every

$500ms$. The simulated and the theoretical results are shown in Fig. 10b. According to (8), we can obtain the relative flow table space estimation error $\zeta = 8.53\%$, which indicates that our model has great universality and can achieve excellent estimation in various scenarios.

Consequently, through comprehensive experiments including the real and the simulated workloads, the simulation results demonstrate that our estimation model can precisely estimate the least required flow table space of all switches in the network, and the relative flow table space estimation error can be less than 10%. In the practical situations, on the basis of our theoretical model, operators can utilize its own historical statistics to estimate the flow table space requirement of the switches at different positions and deploy the most suitable devices. For guaranteeing data plane connectivity, operators can allocate some redundant flow table space for some switches to satisfy the special demands. Moreover, our model can even facilitate topology design and energy assessment.

V. Conclusion

In this paper, we construct a flow table space estimation model based on queuing theory. We divide the life process of the flow table entry into the packet-in process, the handling process and the serving process, and model them as $M/M/1/\infty$, $M/M/1/\infty$ and $M/G/c/c$ systems respectively. Given the upper bound probability of failing to establish the path, we obtain the least required flow table space of every switch. Then we calculate the critical parameters in our model by making some practical assumptions about the traffic distribution and establishing the probability model of routing strategies. The results of comprehensive experiments demonstrate that our model can precisely estimate the least required flow table space of the switches at specific positions, and the relative flow table space estimation error can be less than 10%. Operators can deploy the switches with the proper flow table space in DCNs based on our estimation results and real situations. In the future, we will extend our estimation model to the more general and complicated scenarios.

Acknowledgment

The research is supported by the National Key R&D Program of China under Grant 2017YFB0803202, the project of PCL Future Regional Network Facilities for Large-scale Experiments and Applications, the R&D Program of Shenzhen under grant No. JCYJ20170307153157440 and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

References

[1] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. In *SIGCOMM*, 2013.

[2] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *SIGCOMM*, 2013.

[3] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, 2010.

[4] Andrew R Curtis, Jeffrey C Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: Scaling flow management for high-performance networks. In *SIGCOMM*, 2011.

[5] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[6] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51, 2015.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. In *SIGCOMM*, 2008.

[8] Xuya Jia, Yong Jiang, Zehua Guo, and Zhenwei Wu. Reducing and balancing flow table entries in software-defined networks. In *LCN*, 2016.

[9] Zehua Guo, Yang Xu, Marco Cello, Junjie Zhang, Zicheng Wang, Mingjian Liu, and H. Jonathan Chao. Jumpflow: Reducing flow table usage in software-defined networks. *Computer Networks*, 92:300–315, 2015.

[10] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *IMC*, 2010.

[11] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. Wcmp: Weighted cost multipathing for improved fairness in data centers. In *EuroSys*, 2014.

[12] Naga Katta, Omid Alipourfard, Jennifer Rexford, and David Walker. Cacheflow: Dependency-aware rule-caching for software-defined networks. In *SOSR*, 2016.

[13] Kalapriya Kannan and Subhasis Banerjee. Compact tcam: Flow entry compaction in tcam for power aware sdn. In *ICDCN*, 2013.

[14] Xuya Jia, Qing Li, Yong Jiang, Zehua Guo, and Jie Sun. A low overhead flow-holding algorithm in software-defined networks. *Computer Networks*, 124:170–180, 2017.

[15] Michael Jarschel, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, and Phuoc Tran-Gia. Modeling and performance evaluation of an openflow architecture. In *ITC*, 2011.

[16] Kashif Mahmood, Ameen Chilwan, Olav Østerbø, and Michael Jarschel. Modelling of openflow-based software-defined networks: the multiple node case. *IET Networks*, 4(5):278–284, 2015.

[17] Siamak Azodolmolky, Reza Nejabati, Maryam Pazouki, Philipp Wieder, Ramin Yahyapour, and Dimitra Simeonidou. An analytical model for software defined networking: A network calculus-based approach. In *GLOBECOM*, 2013.

[18] Jie Hu, Chuang Lin, Xiangyang Li, and Jiwei Huang. Scalability of control planes for software defined networks: Modeling and evaluation. In *IWQoS*, 2014.

[19] Bing Xiong, Kun Yang, Jinyuan Zhao, Wei Li, and Keqin Li. Performance evaluation of openflow-based software-defined networks based on queueing model. *Computer Networks*, 102:172–185, 2016.

[20] Yuki Goto, Hiroyuki Masuyama, Bryan Ng, Winston KG Seah, and Yutaka Takahashi. Queueing analysis of software defined network with realistic openflow–based switch model. In *MASCOTS*, 2016.

[21] Zhiruo Cao, Zheng Wang, and Ellen Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM*, 2000.

[22] Marco Chiesa, Guy Kindler, and Michael Schapira. Traffic engineering with equal-cost-multipath: An algorithmic perspective. *IEEE/ACM Transactions on Networking*, 25(2):779–792, 2017.

[23] Zehua Guo, Ruoyan Liu, Yang Xu, Andrey Gushchin, Anwar Walid, and H. Jonathan Chao. STAR: preventing flow-table overflow in software-defined networks. *Computer Networks*, 125:15–25, 2017.

[24] Anders Nygren et al. Openflow switch specification, version 1.3. 4 (protocol version 0× 04), mar. 27, 2014. *Open Networking Foundation.(Part 1 of 2)*, pages 1–84.

[25] Vern Paxson and Sally Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[26] Donald Gross. *Fundamentals of queueing theory*. John Wiley & Sons, 2008.

[27] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *SIGCOMM*, 2015.

[28] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008.