Towards SDN-based Deterministic Networking: Deterministic E2E Delay Case

Aiman Nait Abbou
Aalto University
Espoo, Finland
aiman.naitabbou@aalto.fi

Tarik Taleb
Aalto University and Oulu University
Espoo and Oulu, Finland
tarik.taleb@aalto.fi

JaeSeung Song
Sejong University
Seoul, South Korea
jssong@sejong.ac.kr

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685656

***Abstract*—The explosion of the number of things connected to the Internet gave birth to a new set of services. Customers now are expecting next-generation networks to satisfy vertical applications with different requirements. For instance, 5G networks can carry these various demands by logically dividing the physical network into multiple slices, each slice features specific characteristics based on the type of service. Software-Defined Network (SDN) and Network Function Virtualization (NFV) introduced the term ”Network Softwarization” which is the main enabler of network slicing and 5G networks. Specifically, SDN separates the control plane from the data plane, this concept brings many benefits such as dynamism, flexibility, and innovation. However, when it comes to the assurance of Quality of Service (QoS), SDN is still behind. Not only SDN was not optimized for real-time communications, but also SDN networks cannot offer a deterministic End-to-End (E2E) delay. In this paper, we study OpenFlow-based communications with a focus on the delay. The modeling of queuing delays shows a stable linear development of the mean waiting time under a probability of 0.2 that 10 switches generate *packet_in* message. After that, the increase becomes exponential and thus hard to predict.**

***Index Terms*—SDN, Queuing delay, Delay analysis, Queuing model, Deterministic Networking, 5G, and Beyond 5G.**

I. Introduction

The number of the new Internet of Things (IoT) devices installed is estimated to be 35 Billion by the end of 2021 [1]. Many applications will take advantage of this concept including connected vehicles, smart homes, healthcare, and the industrial sector. Specifically, the industry is believed to benefit the most from IoT as it will provide cost-efficient monitoring and optimized control. The evolution of IoT connecting people, data, and machines as they relate to manufacturing gave birth to the Industrial Internet of Things (IIoT). As a subset of IoT, IIoT covers self-controlling and intelligent Machine-to-Machine (M2M) communications. By the end of 2025, the number of IIoT connections will jump to 36.8 Billion compared to 17.7 back in 2020 [1]. This growth can be explained by the higher rate of deployment due to the cost-effectiveness of the hardware and the ease of accessibility. Regardless of the global COVID-19 pandemic, few Asian manufacturing hubs such as South Korea, Japan, and China have successfully maintained the production despite a lack of manpower. IIoT played a crucial role in filling the void by monitoring the production process, through the use of sensors, increasing automation, and operational efficiency.

Beyond 5G (B5G) is considered as one of the building blocks of smart manufacturing. These future wireless communication systems will enhance efficiency, flexibility, production speed, and diversity during the manufacturing process. The growing investment in 5G and IIoT will contribute to the vision of ”Factories of the future” [2]. The emerging use cases on these factories are diverse with unique characteristics and demands. 3rd Generation Partnership Project (3GPP) 5G identifies three main categories to cover any vertical use case: enhanced Mobile Broadband (eMBB), massive Machine Type Communications (mMTC), and Ultra-Reliable and Low Latency Communications (URLLC) [3]. To accommodate an extended number of heterogeneous vertical services over a single infrastructure, 5G architecture implements a key feature called network slicing (NS). This concept divides the 5G physical network into multiple distinct logical networks (i.e. slices), these slices are isolated, have independent control and management, and support connections with different requirements for latency, reliability, and bandwidth [4].

NS provides efficiency, cost reduction, and flexibility. The two main enablers of this concept are Software-Defined Networking (SDN) and Network-Function Virtualization (NFV). SDN is a softwarization technology that separates the control plane from the data plane. This separation gives the controlling entity (i.e., SDN controller) a global view of the network, enhances flexibility, programmability, innovation, and quick reaction to changing network conditions [5]. Despite all the benefits of this paradigm, SDN is still far from maturity, not only Northbound Application Programming interface (API) is not yet standardized [6], but also SDN was not optimized for real-time communications. Moreover, SDN is not latency-aware and cannot guarantee a bounded End-to-End (E2E) latency. This is mainly because of interactions between the controller and the forwarding devices.

The contributions of this paper include: *i)* OpenFlow network delay breakdown where we take a closer look on the metrics that can impact the E2E delay; *ii)* modelling of OpenFlow-based network components, and numerical analysis of these models. The remaining of this paper is organized as follows: Section II presents the related work. Section III breaks down the network delay, specifically the variable part: processing delay (Section III-A); and queuing delay (Section III-B). Section IV describes the SDN network queuing model, the modelling covers SDN controller in Section IV-B, and SDN switch in Section IV-A. Section V provides a numerical analysis of the network queuing model. Finally, Section VI concludes the paper.

II. Related Work

In our previous work [7], we have proposed Software-Defined Queuing (SDQ). This framework defines queuing algorithms and manages the priority queues on the data forwarding devices. This prototype is based on two main modules. The first one orchestrates the queues based on the incoming traffic. The second module executes the traffic engineering tasks by finding the optimal path and balancing the load. Despite the good overall performance, SDQ does not predict high queuing delays. Also, the service time is not consistent in scenarios with a high number of packets. This work will give us a better understanding of a predictive and deterministic E2E delay, and reach the vision of proactive AI-based decisions on a distributed architecture to ultimately manage multiple slices deployed on different technological domains [8]. Other existing works proposed a queuing model to analyze an OpenFlow–based SDN network. For instance, in [9], the authors tried to push the model to be as realistic as possible. The model takes into account three classes of packets and analysis covering the packet loss probabilities and the average packet delay transfer. Compared to our work, this model considered only one controller and one switch.

Time-Sensitive Networking (TSN) has shown its effectiveness in delivering messages with a guaranteed bounded delay and can dynamically configure the flows [10]. However, managing traffic schedule, bandwidth allocation, and time-synchronization are still open problems in terms of reliability and consistency in critical services. Li *et al.* [11] proposed a network delay model, based on 802.1Qbv switch, with three types of traffic, Best-effort (BE), Stream Reservation (SR), and Control-Data Traffic (CDT). The Forwarding process is managed by Gate Controller List (GCL); this entity decides if a flow is eligible for transmission or not. In order to optimize the resource utilization and calculate the worst-delay by Credit-based Shaper (CBS), the authors proposed a SDN-based network model; this solution was tested with avionics systems as a targeted use case [11]. In this work, SDN was utilized only for bandwidth resource allocation, which is understandable as SDN was not optimized yet for real-time communications. This partnership of TSN and SDN has a lot of potential. But this solution is far from maturity: TSN hardware is still costly and virtual TSN switches are not reliable yet. As a result, offering a deterministic E2E delay on an OpenFlow-based communication is still a relevant objective. Therefore, a full understanding of delay is needed in this type of communications.

III. Network delay breakdown

Data transmission is known to only take place via two adjacent nodes. However, the path from a source to a destination node may include several intermediate hops. The term E2E delay refers to the summation of the delays recorded at each hop on the way to the destination. The delay at each hop can be subdivided into two main parts: firstly, a fixed part combining the Transmission Delay (*TD*) at the sender and the Propagation Delay (*PgD*) over the link; and secondly, a variable part such as the Processing Delay (*PrD*) and Queuing Delay (*QD*) at the sender [12].

A. Processing Delay

The overall processing delay is dependant on the hardware's core processing power and memory access speed. Simple packet forwarding may take few microseconds, but complex operations such as Virtual Private Network (VPN) establishment or data encryption may add 50% to the processing delay. The modelling of this type of delay can be quite simple in traditional networking [13]. However, in SDN networks, data plane devices feature low computation power. As the majority of the heavy processing duties are handled by the controller, the processing delay of new unrecognizable flows can go as much as few seconds. Processing Delay (PrD) is detailed in Fig. 1.

1) The switch extracts the header from the incoming packet and tries to find a match on the flow table. Once the flow packet is matched, the corresponding action will be executed.
2) If no flow entries can match the packet, the packet is sent from the forwarding engine to switch CPU via PCIe bus [14]. This delay will be marked as (*I1*). At a certain point, PCI SDK chip gets the packet and dispatches it to the OpenFlow agent (*I2*). Afterwards, the agent processes the packet, generates a packet_in message and sends it to the controller via a secure OpenFlow channel. This packet contains metadata and the first 128 bytes of the original packet, and this delay will be marked as (*I3*). The period of time of all these three steps can be classified as an inbound latency (*I1-I3*).
3) The SDN controller processes the packet_in message and generates flow_mod and packet_out messages, the delay of this parsing is (*PiP*). Based on the policies configured by the network administrator, the control messages are sent back to the switch through a secure OpenFlow channel (*O1*). Next, the agent on the switch side parses the control message, and translates the logical commands into hardware-based instructions. It includes the addition or removal of the new flow rule (*O2*). Afterwards, depending on the flow rule, PCI SDK chip may require a rearrangement of the previous flow rules to host the new one (*O3*). Finally, the rule is added or removed to the hardware flow table (*O4*). The period of time since flow_mod message is released by the SDN controller, and the update of hardware table can be classified as *outbound latency* (*O1-O4*).

The processing delay is basically the summation of both *inbound* and *outbound latency* [14].

B. Queuing Delay

Queuing delay (QD) can be defined as the waiting time a packet can experience. For instance, in the case of a bursty flow or multiple incoming flows, the node may not be able to process the packets right away. The incoming packets are

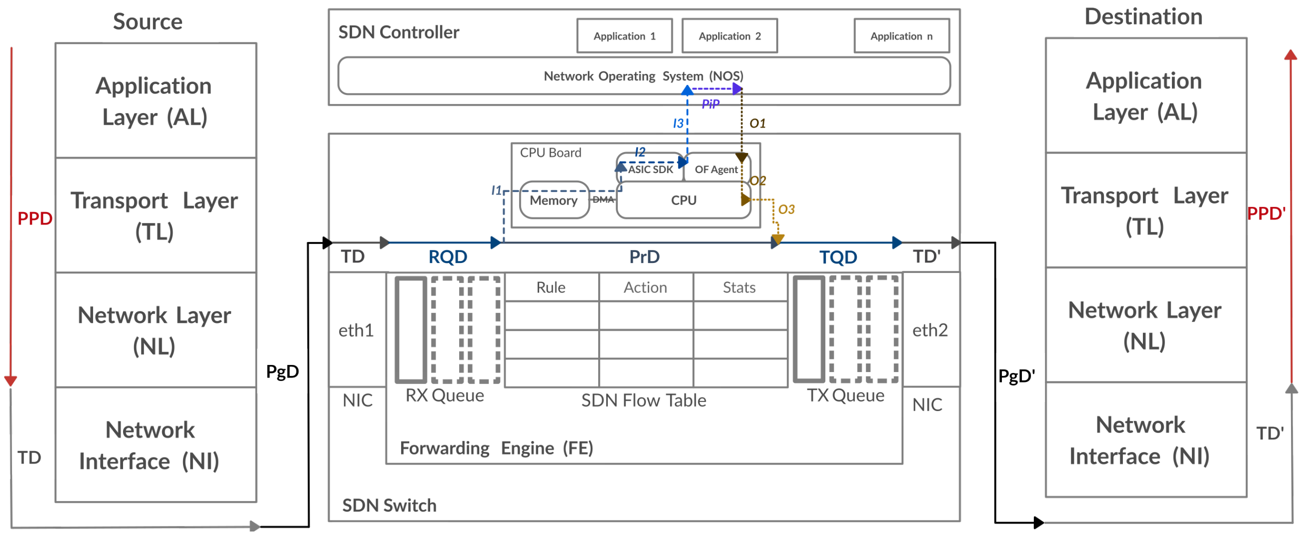


Fig. 1: End-to-End Delay composition in an OpenFlow based network.

enqueued in a buffer. Fig. 2 illustrates an example of a First-In-First-Out (FIFO) concept with two scenarios: (2A) with no competition, thus the incoming packet does not experience extra queuing time, (2B), on the other side, shows two incoming traffic flows at the same time which can increase the waiting time.

IV. Delay modelling

SDN paradigm separates the control plane from the data plane. The communication between the planes is the main reason for extra waiting time. The SDN controller communicates with the forwarding devices through a southbound interface (e.g., OpenFlow). This section will provide a modelling of SDN controller and SDN switches as queuing systems. First, the SDN switch will be modeled as an $M/H_2/1$ queue, whereas the controller is represented as a basic $M/M/1$ *queue*.

A. *Queuing model: SDN switch*

OpenFlow switch on SDN network can be represented as a single server queue with two-phase nodes ($M/H_2/1$). We assume that the packet arrival process follows a Poisson process with rate λ_s, where s is the switch identifier. There is

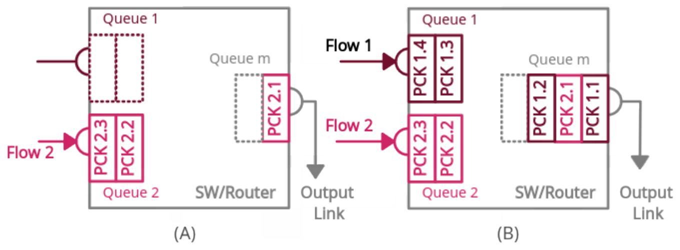


Fig. 2: Basic FIFO queuing, (A): One incoming flow, (B): Two flows at the same time.

a probability of α that the incoming packet does not feature any flow rule on the table (after sending a packet-in message to the controller). Thus, the probability of finding a match on the table is $(1-\alpha)$. The service time on each switch can be represented as two-phase hyper-exponential distribution. With probability α the packet receives service at rate μ_1, while with probability $(1-\alpha)$ this packet receives service at rate μ_2. As as shown in Fig. 3, the state of the system is represented by the pair (n, i) where n is the total number of packets in the queue, and i is the current service phase (i can be only 1 or 2).

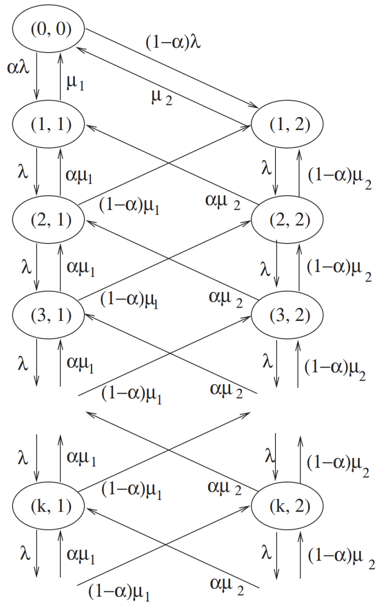


Fig. 3: State transition diagram of an $M/H_2/1$ queue.

To find the service time for each packet that enters the switch, let us consider the $M/H_2/1$ Queuing System with the packet arrival density function described in (1) and the service-time density function (2)

$$a(t) = \lambda e^{-\lambda t} \tag{1}$$

$$d(t) = \alpha \mu_1 e^{-\mu_1 t} + (1-\alpha)\mu_2 e^{-\mu_2 t} \tag{2}$$

The Laplace integral (transform) of Functions (1) and (2) is shown in (3) and (4), respectively.

$$A^*(s) = \frac{\lambda}{s+\lambda'} \tag{3}$$

$$D^*(s) = \alpha \frac{\mu_1}{s+\mu_1} + (1-\alpha)\frac{\mu_2}{s+\mu_2} \tag{4}$$

The spectral decomposition can be expressed as shown in (5)

$$A^*(-s)D^*(s) - 1 = \frac{\phi_1(s)}{\phi_2(s)} \tag{5}$$

Based on the distributions (1) and (2), and considering the conditions (3) and (4), the spectral decomposition can be described as (6):

$$\begin{aligned}\frac{\phi_1(s)}{\phi_2(s)} &= \frac{s\left(s^2+c_1 s+c_0\right)}{(\lambda-s)(\mu_1+s)(\mu_2+s)} \\ &= \frac{s\left[s+\left(\frac{c_1}{2}-\sqrt{\frac{c_1^2}{4}-c_0}\right)\right]\left[s+\left(\frac{c_1}{2}+\sqrt{\frac{c_1^2}{4}-c_0}\right)\right]}{(\lambda-s)(\mu_1+s)(\mu_2+s)} \\ &= \frac{s(s+r_1)(s+r_2)}{(\lambda-s)(\mu_1+s)(\mu_2+s)}\end{aligned} \tag{6}$$

where $c_0 = \mu_1\mu_2 - \lambda\left[(1-\alpha)\mu_1 + \alpha\mu_2\right]$ and $c_1 = \mu_1 + \mu_2 - \lambda$ are coefficients of the polynomial $s^2 + c_1 s + c_0$. The mean waiting time, which is average service time the packet spends on the switch, is:

$$\overline{W} = \frac{r_1+r_2}{r_1 r_2} - \frac{\mu_1+\mu_2}{\mu_1\mu_2} \tag{7}$$

According to Vieta's formula, the relationship between coefficients c_0 and c_1 and the roots r_1 and r_1 is:

$$\begin{aligned}c_1 &= r_1 + r_2 = \mu_1 + \mu_2 - \lambda \\ c_0 &= r_1 r_2 = \mu_1\mu_2 - \lambda\left[(1-p)\mu_1 + p\mu_2\right]\end{aligned} \tag{8}$$

From (7), (8), the expression of the waiting time is:

$$\begin{aligned}\overline{W} &= \frac{r_1+r_2}{r_1 r_2} - \frac{\mu_1+\mu_2}{\mu_1\mu_2} \\ &= \frac{\mu_1+\mu_2-\lambda}{\mu_1\mu_2 - \lambda\left[(1-\alpha)\mu_1+\alpha\mu_2\right]} - \frac{\mu_1+\mu_2}{\mu_1\mu_2} \\ &= \frac{\lambda\left[(1-\alpha)\mu_1^2+\alpha\mu_2^2\right]}{\mu_1\mu_2\left[\mu_1\mu_2-\lambda\left[(1-\alpha)\mu_1+\alpha\mu_2\right]\right]}\end{aligned} \tag{9}$$

Based on (9), and Little's law formula, the mean number of $packet\_in$ messages on the switch is:

$$\overline{L} = \lambda \overline{W} \tag{10}$$

B. Queuing model: SDN controller

Birth-Death Processes are special continuous-time Markov chains [15], indexed by integers, whereby the transitions are only allowed with the closest neighbours $(j+1)$ or $(j-1)$ for a state $j > 0$ and the state 1 when the system is empty (state $j = 0$). SDN controller can be modelled as an $M/M/1$ $queue$, which is one of the simplest queues. The arrival process follows a Poisson process whereas the service time is exponentially distributed. A packet_in message, entering the controller with a rate λ, is identified with a *birth*, whilst the processed packet, leaving the controller back to the switch with a rate μ, is referred to as a *death*, as shown in Fig. (4).

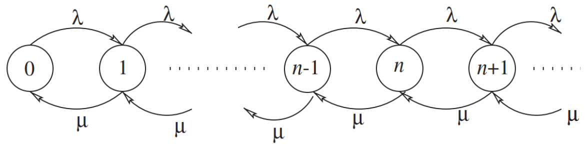


Fig. 4: State transition diagram of an $M/M/1$ queue.

Multiple switches can be associated with an SDN controller. The packets arrive at a switch i following a Poisson process. If a match is found, the waiting time only depends on the switch performance, otherwise, there is a probability of ρi that the switch i generates a packet_in message. The SDN controller receives the packet_in messages and processes them by order of arrival. Let s denote the number of switches on the SDN network affiliated to the same SDN controller. Let assume the packet arrival rate to the switch i follows a Poisson process with rate λ_i. The controller receives packet_in messages from switch i with a probability of ρ_i. Therefore, we have

$$\lambda = \sum_{i=1}^{s} \lambda_i \rho_i \tag{11}$$

Let N be a random variable that represents the number of messages on the controller at a steady state. The probability that at a steady-state the number of packets present in the controller is n can be denoted by p_n.

$$\begin{aligned}p_n &= \text{Prob}\{N = n\} \\ &= \rho^n(1-\rho) \quad for \quad \rho = \frac{\lambda}{\mu} < 1\end{aligned} \tag{12}$$

Thus, the average number of packets in the controller at steady state is:

$$\begin{aligned}L = E[N] &= \sum_{i=0}^{\infty} n p_n = \sum_{n=0}^{\infty} n(1-\rho)\rho^n \\ &= (1-\rho)\sum_{n=0}^{\infty} n\rho^n = (1-\rho)\rho\sum_{n=0}^{\infty} n\rho^{n-1}\end{aligned} \tag{13}$$

If we consider the controller as a stable system, the utilization factor $\rho < 1$, the expression of the mean number of packets is:

$$L = E[N] = (1-\rho)\frac{\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda} \tag{14}$$

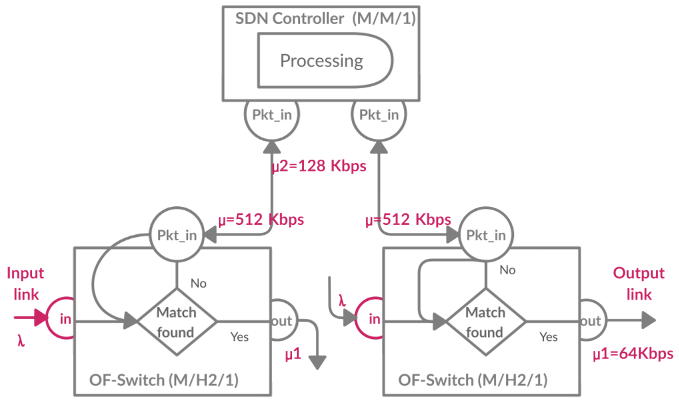


Fig. 5: The interaction betweeen the two-phase system (SDN switch) and birth–death process system (SDN controller).

Let R denote the random variable that describes the response time of the packets inside the controller. Little's law formula (15) defines the relationship between $E[N]$ and $E[R]$, the mean waiting time the packets spent at the controller, as follows:

$$\begin{aligned} E[R] &= \frac{1}{\lambda}E[N] \qquad (W = \frac{1}{\lambda}L) \\ &= \frac{1}{\mu - \lambda} = \frac{1}{\mu - \sum_{i=1}^{s} \lambda_i \rho_i} \end{aligned} \tag{15}$$

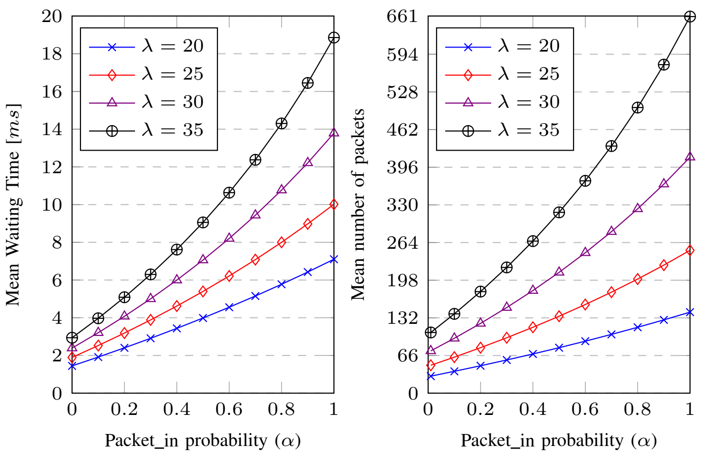
V. Performance Evaluation

Assuming every SDN switch is receiving the packets with a fixed rate λ, then the packets are either sent to the output port with a rate of μ_1, i.e., a matching flow entry is found, or the $packet\_in$ is sent to the controller with a rate of μ_2. The controller then processes the packet and sends it back to the switch with a rate of μ as shown in Fig. 5.

A. Numerical analysis: SDN switch

Fig. 6 illustrates the performance of a SDN switch, specifically, the mean Waiting Time (WT) as shown in Fig. 6a, and the mean Number of $Packet\_in$ (NP) messages (Fig. 6b) per probability α. Overall, WT and NP in the switch increase with the probability of not finding a match. It is noticeable that the degree of growth is more significant with higher data rates. For instance, in low data rates (e.g., 20 and 25 Kbps), the expansion of both parameters is linear, also the value of the service time and the number of packets are just under 2 ms and 49 $packet\_in$, respectively, when the probability α is close to zero. When the probability is close to 1, the peak average reaches 7.1 ms and 142 $packet\_in$ in the switch. This case is quite rare, i.e., all the incoming packets are sent to the controller for a decision. However, this is possible especially during the first deployment of the network or after adding new forwarding devices where the flow tables are empty.

On the other side, in higher data rates (e.g., 30 and 35 kbps), WT and NP are slightly under 3 ms and 107, respectively. Furthermore, the expansion is linear when the probability is less than 0.2, beyond that, both parameters surge, and the growth becomes exponential. As a result, the two parameters reach a ceiling average of 19 ms and 660 packets when the probability is close to 1. To conclude the switch queue analysis, the probability of not finding a match α can dictate the average service time and the number of $packet\_in$ messages. Moreover, less amount of data entering the switch (λ) can keep WT and NP under control. Eventually, fewer interactions with the controller will result in lower E2E latency (i.e., proactive forwarding).



(a) Mean Waiting time of $packet\_in$ messages per probability α.

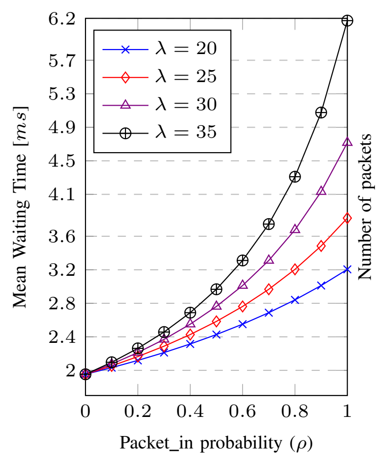
(b) Mean number of $packet\_in$ messages per probability α.

Fig. 6: Performance evaluation of an SDN switch based on $M/H_2/1$ queue.

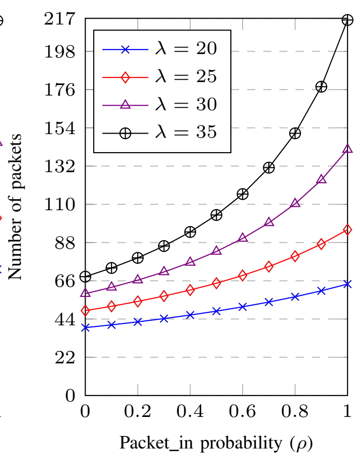
B. Numerical analysis: SDN controller

When it comes to the SDN controller, not only the probability of receiving the packets from a switch matters, but also the size of the SDN network. Therefore, fewer switches associated with the controller translates to less $packet\_in$ messages to be processed. We evaluate the SDN controller's queue either by fixing the number of switches at 10 (Fig. 7) or by fixing the probability ρ at 0.05 (Fig. 8). In the case of 10 switches, the Average Waiting Time (WT) (shown in Fig. 8a) starts from just under 2 ms regardless of the data rate (λ) when the probability ρ is close to zero. In low data rates, WT slowly rises with the probability α. It reaches 3.2 and 3.8 ms for 20 and 25 $kbps$. In higher data rates, the WT surges and can reach three times as the probability gets closer to 1, i.e., 4.7 and 6.2 ms for 30 and 35 $kbps$. Similarly, when we fix the probability at 0.05 and we increase the number of switches (Fig. 7a), we get similar results for a higher number of switches. This means that both probability ρ or the NP has an impact on the number of $packet\_in$ messages sent to the controller.

Similarly to the WT, the expansion of the mean NP inside the controller is slow in low data rates (Fig. 7b), but in high data rates, the NP surges quickly after a probability of 0.2 and can reach up to 142 and 216 for 30 and 35 $Kbps$, respectively. To conclude this analysis, the more $packet\_in$ messages the controller has to process (i.e., reactive forwarding), the longer the waiting times of the packets in the system. In summary, the growth of the mean waiting time is linear to the probability



(a) Mean Waiting time of $packet\_in$ messages per probability α.

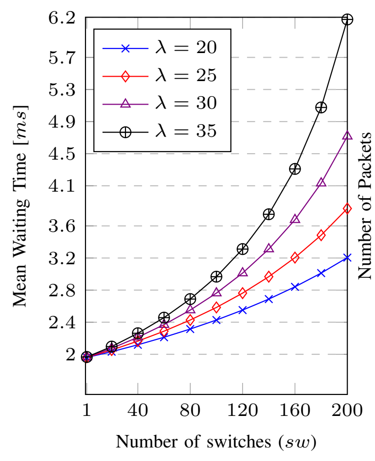


(b) Mean number of $packet\_in$ messages per probability α.

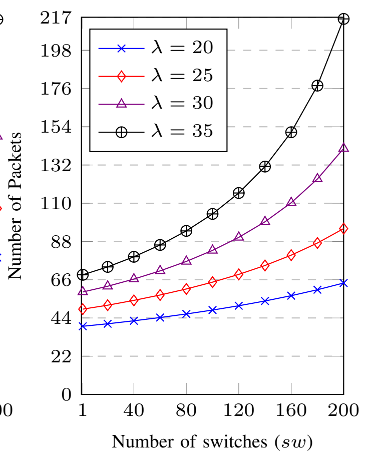
Fig. 7: Performance evaluation of an SDN controller ($M/M/1$ $queue$) managing 10 switches.

if $\rho \leq 0.2$ (Fig. 7a), and to the number of switches if $sw \leq 40$ less than 40 (Fig. 8a). Therefore, we can expect a stable system in terms of average queuing time only when the following bounded inequality is verified:

$$0 \leq \rho \times sw \leq 2 \tag{16}$$



(a) Mean Waiting time of $packet\_in$ messages per number of switches (sw).



(b) Mean number of $packet\_in$ messages per Number of switches (sw).

Fig. 8: Performance evaluation of an SDN controller ($M/M/1$ $queue$) with a fixed $packet\_in$ probablity of 0.05.

VI. Conclusion

To conclude, this paper provided a breakdown of the network delay in an Open-Flow based communication network. In order to achieve a deterministic E2E delay, the queuing and processing delay at each hop need to be controlled. In this paper, we provided a queuing model for SDN controller and SDN switches. Based on the numerical analysis, the growth of the average queuing delay in both systems (i.e., controller and switches) is related to the probability of a switch sending a $packet\_in$ message to the controller for a decision. Moreover, in our scenario, the average waiting time of a controller associated to 10 switches can be linear and under control with a probability under 0.2 of receiving a $packet\_in$ message. The same situation happens with a probability of less than 0.05 and 40 switches . Nevertheless, time-sensitive applications cannot afford waiting for a flow rule to be installed. The controller should be able to proactively install the flow rules before the arrival of the packets. Our next work will focus on the automation of this process and the prediction of the E2E delay based on the queuing and processing delays.

Acknowledgment

This work was partially supported by the European Unions Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant No. 871780. It was also supported in part by the Academy of Finland 6Genesis project under Grant No. 318927 and IDEA-MILL with grant number 33593. Prof. Song was supported by the Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (Grant No. 2021-0-00188 and IITP-2021-0-01816).

References

[1] "Industrial iot: 2020-2025," juniperresearch.com/researchstore/devices-technology/industrial-iot-market-research, accessed: 2021-04-10.

[2] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5g network slices to support industrial internet and to shape next-generation smart factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[3] J. Prados-Garzon and T. Taleb, "Asynchronous time-sensitive networking for 5g backhauling," *IEEE Network*, vol. 35, no. 2, pp. 144–151, 2021.

[4] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward using reinforcement learning for trigger selection in network slice mobility," *IEEE Journal on Selected Areas in Communications*, 2021.

[5] Z. Shu and T. Taleb, "A novel qos framework for network slicing in 5g and beyond networks based on sdn and nfv," *IEEE Network*, 2020.

[6] E. P. Neto, F. S. D. Silva, L. M. Schneider, A. V. Neto, and R. Immich, "Seamless mano of multi-vendor sdn controllers across federated multi-domains," *Computer Networks*, vol. 186, p. 107752, 2021.

[7] A. N. Abbou, T. Taleb, and J. Song, "A software-defined queuing framework for qos provisioning in 5g and beyond mobile systems," *IEEE Network*, vol. 35, no. 2, pp. 168–173, 2021.

[8] B. Chafika, T. Taleb, C.-T. Phan, C. Tselios, and G. Tsolis, "Distributed ai-based security for massive numbers of network slices in 5g & beyond mobile systems," in *EuCNC/6G Summit*. IEEE, 2021, pp. 401–406.

[9] Y. Goto, B. Ng, W. K. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic openflow–based switch model," *Computer Networks*, vol. 164, p. 106892, 2019.

[10] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Learnet: Reinforcement learning based flow scheduling for asynchronous deterministic networks," in *ICC 2020-2020*. IEEE, 2020, pp. 1–6.

[11] E. Li, F. He, L. Zhao, and X. Zhou, "A sdn-based traffic bandwidth allocation method for time sensitive networking in avionics," in *IEEE/AIAA 38th DASC*. IEEE, 2019, pp. 1–7.

[12] M. Yang, X. R. Li, H. Chen, and N. S. Rao, "Predicting internet end-to-end delay: an overview," in *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*. IEEE, 2004, pp. 210–214.

[13] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *GLOBECOM'04*, vol. 3. IEEE, 2004, pp. 1629–1634.

[14] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E. Li, and M. Thottan, "Measuring control plane latency in sdn-enabled switches," in *1st ACM SIGCOMM*, 2015, pp. 1–6.

[15] W. J. Stewart, *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton university press, 2009.