Computers and Electrical Engineering 81 (2020) 106546

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Handover analysis of openflow-based mobile networks with distributed control plane ☆

Strahil Panev [a,*], Pero Latkoski [b]

[a] Ericsson Telecommunications Macedonia, Skopje, North Macedonia
[b] Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, North Macedonia

ARTICLE INFO

Article history:
Received 30 July 2019
Revised 24 December 2019
Accepted 26 December 2019
Available online 2 January 2020

Queueing model
SDN
OpenFlow
Performance evaluation
Mobile networks

ABSTRACT

A centralized controller approach in Software-Defined Networking (SDN) creates problems related to scalability and performance degradation. These problems are the main reason for introducing an architecture of logically distributed controllers. Ensuring the continuity of a user's ongoing session and reducing the interruption time during a handover, is one of the major challenges in today's modern mobile networks. In this paper, we propose a novel analytical approach to model the delay introduced by the handover-related OpenFlow signaling messages in SDN networks using multiple controllers. The analytical model is verified using extensive simulations. The results show that high probability of *Packet-in* messages causes rapid degradation of network performance. Similarly, by increasing the number of controllers, the number of synchronization messages increases, thus impacting the packet service time. Our findings can facilitate the design of delay-constrained handover approaches in a mobile network with a target system throughput.

1. Introduction

SDN has been receiving an enormous attention from both the industry and academia in recent years. It is an emerging technology that fully separates the control and user plane, enabling network abstraction, and programmability via open standard protocols. SDN is primarily used with layers 2–4 of the Open Systems Interconnection (OSI) model and other protocols such as Multiprotocol Label Switching (MPLS) [1]. The outcome of using SDN is two-fold: (1) logically centralized network intelligence and global state management; and (2) separate underlying infrastructure that is completely decoupled from network applications. The key benefit offered by SDN is programmability and network abstraction. It enables a complex network consisting of many individual devices to be programmed and orchestrated as a single entity. Therefore, a SDN Controller (SDNC) and its respective interfaces (comprising the control plane) can act as a centralized intelligence with a global overview of the network and the authority for networking reconfiguration.

Nowadays, OpenFlow (OF) is the most widely spread protocol for controller-to-switch communication. Every OF switch contains one or several flow tables that perform packet lookups and forwarding [2]. At the initialization of the network, or during topology updates, faults etc., the controller can add, update, delete the flow entries in the flow tables, and this can be done reactively (as a reply to the switch packets) or proactively. Each flow table contains a set of flow entries, and each flow entry consists of match fields, counters, and a set of instructions that are applied when matching the incoming

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the editor-in-chief by associate editor Dr. Vyasa Sai.

* Corresponding author.
E-mail address: strahil.panev@ericsson.com (S. Panev).

https://doi.org/10.1016/j.compeleceng.2019.106546
0045-7906/

packets. The matching of packets starts with the first flow table and can continue sequentially to the other flow tables. When a matching entry is found, the specific instructions associated with that flow are executed. In an OF-network, if an incoming flow cannot be matched to a specific forwarding instruction, the following actions are executed: (i) the first packet of the flow is sent from the switch to the controller; (ii) the controller computes the forwarding path and sends instructions to the involved switches in the data path with specific information for the flow entries to be added in the flow tables; (iii) all the following packets of the same flow are forwarded to the next hop by using the information already provided by the controller and there is no need of any additional control plane action. An OF-based network can have single or multiple controllers. In this paper, we analyze a SDN network with a distributed control plane (multiple controllers).

Besides the advantages, there are some intrinsic disadvantages to having a centralizing network intelligence in a single entity, such as a higher probability of performance and scalability issues, and an introduction of a single point of failure in the network [3]. Moreover, since mobile networks handle many users, a single controller may not be able to serve all requests coming from all the switches within a reasonable time. The aforementioned challenges are the main reason for deploying multiple controllers, and having switches belonging to domains, whereby each switch is only managed by a single controller that is responsible for the domain. This distributed network created by the controllers has no hierarchical structure, thus there is no unique global network view, and the controllers need to exchange network information [4]. In a distributed SDN architecture, a single controller exchanges messages with the application (using the Northbound interface), with other controllers (using the West and East bound interfaces) and with the data plane (through the Southbound interface). Although multiple controllers can significantly contribute to the network performance, this approach increases the signaling overhead, requires additional computing resources, and introduces higher complexity in the design of the network. To summarize, introducing additional controllers in the network, reduces the amount of per controller traffic, however the exchange of signaling messages between the controllers becomes higher. For the purpose of our analysis, it is important to estimate the impact on the handover delay in the case of deploying multiple controllers.

Nowadays, the SDN concept is already massively used for data centers, mobile backhaul, and core networks, providing mobile operators faster time to market, innovative services, decreased capital and operational costs, and efficient use of network resources. Due to the extensive growth of mobile data traffic, there is a serious concern regarding provisioning high Quality of Service (QoS) and meeting requirements of the demanding mobile users. A major challenge in today's mobile/cellular networks is ensuring the continuity of the user's ongoing session and providing minimal interruption time during the handover procedure. In the case of SDN-based core networks, a successful handover procedure consists of exchanging OF signaling messages between the control and data plane. The specific case of "hard" handover involves a breakage of the ongoing session and requires reconfiguration and management messages to be exchanged between the switches and the controller. Our work focuses on the handover of hosts between different switches, and our aim is to model the handover delay introduced by the OF control messages by using a novel mathematical approach.

Analytical models based on queuing theory have been extensively used to provide performance evaluation of systems based on OF architecture [5]. Analyzing particular parameters for an OF network under specific conditions can lead to novel algorithms and valuable lessons learned regarding network performance. In the literature, there are several existing analytical models [6], however, not much work has been done related to using multiple controllers, and/or using different probability distributions for modeling the switch/controller with the exception of the M/M/1 queue. In this work, we propose a novel analytical approach for modeling the handover related messages in the case of a Mobile Node (MN) moving from one switch to another in a network with multiple controllers. Further, we analyze the key factors that have major influence on the controllers' packet service time, such as *Packet-in* and synchronization message arrival rates, number of network elements, controller load, and system throughput. Our model is later verified through extensive simulations. To the best of our knowledge, no previous work has addressed the modeling of handover delay in a distributed controller mobile network.

The main contributions of this work are summarized as follows:

(1) We propose a novel analytical approach to model the performance of the switches and the multiple controllers based on queueing theory. The switches and the controllers are modelled by using M/M/1 and M/PH/1 queue, respectively, and a suitable performance metric for assessing the handover delay is proposed. We expand previous related articles by modeling not only *Packet-in* messages, but also *Port-status* messages. *Port-status* messages are of key interest for "hard" handover and are the first messages being exchanged in the control plane, later followed by a specific sequence of switch-controller-switch communication.
(2) We validate our analytical model by comparing numerical results against simulation results obtained with our discrete event simulator. The findings can be used when designing handover delay guarantees in a mobile network for a given amount of traffic directed towards the controller. In addition, we analyze key factors, such as the impact of the number of controllers and switches, arrival rates of *Packet-in* and controller synchronization messages, and system throughput. The obtained results prove that the aforementioned aspects are critical factors required for modeling SDN-based networks.
(3) In contrast to the recent efforts based on OF modifications/expansions for mobility management mechanisms targeting a reduced complexity and lower handover latency (compared to the existing non-OF protocols), our proposed mathematical approach can be easily used and extended to allow for similar complementing conclusions.
(4) The remainder of this paper is organized as follows: Section 2 introduces the related work. In Section 3, we describe our network model and elaborate the proposed analytical approach. Section 4 describes the detailed performance

evaluation of the mathematical model and provides a comparison with the results obtained via simulations. In Section 5, we analyze the impact of the modeling assumptions, and finally we state our conclusions in Section 6.

2. Related work

In this section, we investigate similar work done by the research community. We divide our investigation into three major groups: recent papers providing different mathematical approaches with simulation or test-bed validation, proposals using multiple SDN controllers including controller placement and flow balancing, and analysis of contributions in the area of mobility management related to latency and OF investigations.

2.1. Analytical modeling

One of the pioneering works related to modeling of OF-based interaction between a switch and a controller is presented by Jarchel et al. [7]. The authors design an OF test-bed and select performance parameters with aim to introduce an analytical model. They use a feedback-oriented queueing model, which has a forwarding queue system (switch), and a feedback queue system (controller). The proposed model uses M/M queue (infinite queue size) for the switch, and M/M – S (finite queue size) for the feedback queue. The major limitation of this approach is the scenario that involves modeling of a single switch interacting with a single controller.

Sarkar and Setua [6], use the Jackson network for data plane modeling, but implement modifications to better shape the nature of the traffic flow in SDN networks. They also consider a single-node model and deploy a feedback interaction between the switch and the controller. A Poisson distribution for the arrival rate and exponential service rate at both switch and the controller are assumed in this work. In the performance evaluation, the authors measure the average packet service time and the distribution of time spent by each packet. However, they only analyze a single controller/switch and consider only the M/M/1 queueing model. Thieme [8] investigates different aspects and challenges of using queueing theory for performance analysis of SDN. The author is mainly interested in the latencies that occur when moving data between different caches, software and hardware triggered interruptions, design of interfaces as queues, etc. An important subject of analysis is the finite capacity of the system and the limitation on the buffer size, however, as the author explains, most if not all recent work has completely omitted the last two assumptions and considered both as infinite.

The work by Mahmood et al. [5] is valuable as it is the first to considers a multiple-node case, meaning multiple switches constituting the data plane. Their work is extended in [9] with modeling of *Packet-in* messages. The authors propose an analytical approach to quantify the controller's sojourn time and the solidity of the packets pumped into the network. Although the authors do not analyze the propagation delay, their model considers both finite and infinite buffers. Shang and Wolter [10] are also observing the *Packet-in* messages and their impact on the performance of the controller's sojourn time. They use M/H_2 (2-phase hyper exponential distribution) queue to model the switches while modeling the controller as M/M. The authors do not provide simulation validation, but their analytical model clearly points that if the probability of the *Packet-in* messages is high, the performance of the network rapidly degrades.

2.2. Multiple SDN controllers

The limitations of a single controller due to the limited capacity and inability to handle large number of flows, were recently investigated by the research community. Blial et al. [11] analyze multi-controller architectures in SDN-enabled networks. The authors' detailed investigation on the differences between multiple types of architectures includes description of the design, the communication process, and performance results. In general, a multi-controller architecture follows a flat or a hierarchical architecture. In a flat architecture, all controllers are positioned horizontally on a single level, meaning that each controller only has a partial view of its network, while in a hierarchical architecture, the controllers are positioned vertically, portioned among multiple levels. Additionally, the authors survey existing logically centralized architectures (HyperFlow, ONOS, DISCO), and logically distributed architectures (KANDOO, ORION).

Sood et al. [12] describe the inter-dependency of the QoS requirements, the number of controllers, resources, and operational cost minimization. They model their system as a M/M/m queue and propose a solid mathematical approach that allows for controller service time minimization, and evaluation of the impact on the performance of the critical factors, such as number of controllers, number of switches, and cost of resources. Finally, the authors propose a distributed decision based low-balancing scheme, by exploring the economic relationship between the flow arrival rate, resources and cost. Chen et al. [13] investigate the design of a master-slave OF controller (OF-C) arrangement, applicable for software defined elastic optical networks (SD-EON). The authors develop a controller communication protocol (CCP), which allows the master OF-C to synchronize the status with the slave OF-C in real time. The slave can quickly detect the master controller failure and take over the control, so any service disruption is avoided. The proposed framework is implemented in a SD-EON control plane test-bed consisting of high-performance servers used to evaluate the effectiveness of the solution in different scenarios related to network failures.

Heller et al. [14] propose an architecture that adopts cluster of multiple controllers with focus on the exchanged signaling traffic between controllers. The authors develop an analytical model to quantify the processing delay experienced at the switches due to inter-controller communication. By investigating some real network topologies, the authors analyze

the controller placement problem and the possible delay dependencies and trade-offs. Similarly, Hu et al. [15] address the problem of controller placement with the aim of maximizing the reliability of the control network. The authors estimate the impact of the number of controllers on the reliability of the control plane by using real network topologies. By proposing several algorithms for controller placement, the following conclusion is drawn: placing too few or too many controllers can have a severe impact on the reliability. Simulations clearly show the trade-offs between the chosen metrics. Finally, Zhihao et al. [4] investigate an approach to reduce the packet sojourn time by deploying multiple controllers. Their main aim is to evaluate the flow setup time and optimize the number of the controllers. An analytical model is proposed, and a prototype of multiple controllers is implemented. By measuring the packet service time, the authors fit a hyper-Erlang distribution to the response time, which they later use in their mathematical model to determine the optimal number of controllers. Our own work is substantially motivated by the results and the approach proposed by Zhihao et al. [4].

2.3. Mobility management

In [16], Marquezan et al. investigate the key factors that impact the overall latency in SDN-based mobile core networks. They argue that latency is contributed mainly by two factors: processing delay and transmission delay. Processing delay refers to the packet service time within the switch and the controller, whereas transmission delay describes the time needed for the data to be transmitted between the different network entities. The authors assess the impact of the number of hops, load of the controller, and the buffer dimensioning. Duan et al. [17] explore the average packet delay and define the following two types: (i) delay introduced by the switch if there is a need to forward the packet to the controller, (ii) delay that occurs after the handover decision, as both involved switches need to communicate to the controller to indicate a successful handover. They propose a detailed analytical framework to model the packet delay in an OF-based mobile network with a single controller. However, their interest is only the single-node model, and the exchange of signaling messages between the switch and the controller is not considered.

Further, we investigate the recently proposed OF-based procedures to provide a seamless handover in various mobile networks with different radio access. Meneses et al. [18] propose a 5G mobility process optimization framework, where SDN reaches the end device. They modify and extend the standard OF protocol to add mobility management capabilities such that the mobile terminal can provide information to the network about connectivity targets and conditions. Furthermore, this approach enables an enhanced controller that has the capability to enforce the necessary changes back to the mobile node. The MN uses the *Packet-in* message to indicate a decrease in signal quality to the controller, and when the message reaches the controller, the network acts on the received information and makes a handover decision to move the MN from Access Point 1 (AP1) to AP2, by sending a *Flow_mod* message. Their framework mimics a “make-before-break” approach via the first *Flow_mod* message by preemptively implementing a rule that enables the MN to forward packets towards AP2, whereas the second *Flow_mod* message changes the source and destination Medium Access Control (MAC) addresses to fulfill the wireless handshake. The proposed concept is validated with a prototype deployed over a physical wireless test-bed, proving that the handover procedure can be performed completely with OF and without using any additional mobility protocols.

Alotaibi et al. [19] are interested in quantifying the handover delay due to OF-signaling message exchange. They model both *Packet-in* and *Port-status* messages and define a proper performance metric to evaluate the total delay that MN experiences in the case of a “hard handover”. Furthermore, they propose a LTE architecture and compare it with the existing solutions by using extensive simulations. A major drawback in their proposal is the fact that they only model a single controller without assessing the impact on the handover delay in case of a distributed control plane. Moreover, in their numerical results, the authors present findings obtained via simulations in relation to the proposed distributed LTE architecture without focus on extensive validation of the proposed analytical model.

In this context, our paper describes an analytical model for SDN-based mobile networks with a distributed control plane. We focus on host-oriented handover between different switches, and we attempt to quantify the handover latency by investigating the delay caused by the exchange of the OF control plane messages. In the next section, we describe our analytical model.

3. Analytical model

This section contains our proposed analytical model used to quantify the handover delay in SDN-based mobile networks. Initially, we investigate how our model fits into the existing Third Generation Partnership Project (3GPP) mobile architectures in conjunction with the system model overview and the model assumptions. Afterwards, we model a single controller domain, with a focus to design the relationship of the total packet arrival rate and the switch-to-controller messages. Later, we expand our queuing model to approximate the controller’s sojourn time and use it to quantify the total handover delay. The sojourn time is the total time a packet spends at the controller including the time required for the packet to be serviced.

3.1. Applicability of the proposed model in existing mobile networks

We analyze a similar scenario as in [20], depicted in Fig. 1. The approach is called SDN-based Evolved Packet Core (EPC) with partial virtualization. In this way, only the control plane 3GPP applications are virtualized, whereas the user plane is implemented on physical OF-switches. This approach is feasible since the control plane functions have lesser requirements in

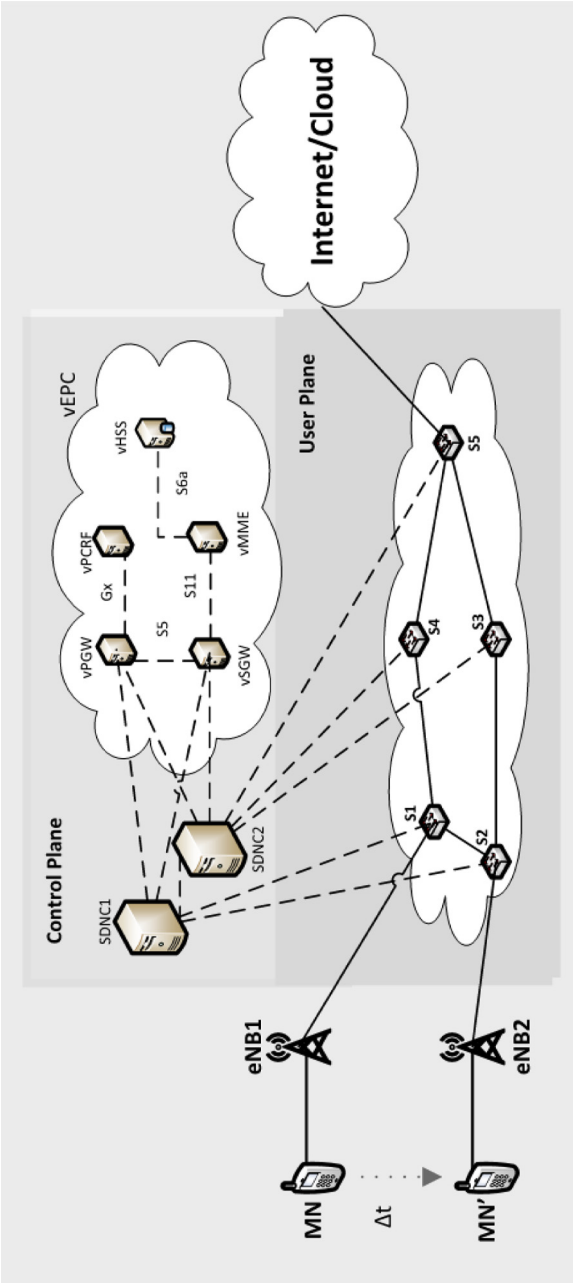

Fig. 1. Framing the proposed model in the LTE architecture.

terms of throughput, but strict requirements on latency and computation. Contrarily, user plane nodes must achieve a very high throughput. In a virtual EPC, the Mobility Management Entity (MME), Home Subscriber Server (HSS) and Policy and Charging Rules Function (PCRF) are pure Control Plane (CP) functions. The control software of Packet Data Network Gateway (PGW) and Serving Gateway (SGW) is logically centralized, and via the north-bound Application Programming Interface (API) runs on top of an OF-SDNC, as an application. The User Plane (UP) functions of SGW and PGW are installed over SDN switches and controlled by SDNC on the southbound API.

We analyze a scenario where due to MN's mobility, a handover is needed from an evolved Node B1 (eNB1) to eNB2. This paper does not focus on the Radio Access Network (RAN) impact, or the 3GPP mobility management protocols. Instead, we investigate the possibility and try to quantify the impact that the handover procedure has on the latency, when the standard OF protocol is used for signaling between the network servers. If we want to incorporate the impact of the eNB on the handover latency, we can easily simplify this impact and extend the model by considering the latency introduced between the MN and the eNB. This latency is mainly to the physical layer, and it comprises of: time to transmit, processing time at both MN and the eNB, propagation delay and retransmissions. The processing delay at MN involves code block segmentation, channel coding, rate matching, multiplexing, channel interleaving, etc. The processing time in the eNB is attributed to channel coding, scrambling, OFDM signal generation, cyclic redundancy check, rate matching, etc. However, the latency between the MN and eNB is negligible compared to the overall handover delay that the MN is experiencing when performing a "hard" handover.

3.2. System description and assumptions

Our aim is to model controllers' service time in relation to the packet arrival rate at switch level (system throughput). The network is comprised of OF switches and multiple controllers. When a packet is received at the switch interface, the switch initially tries to match the packet header to a flow entry, otherwise it sends a *Packet-in* message to the controller. When a MN disconnects or connects from/to a switch, it sends a *Port-status* message. In OF v.1.3.1 [2], the switch may initiate asynchronous messages (*Packet-in, Port-status, Flow-mod/removed, Error*) or symmetric messages (*Hello, Echo, Experimenter*). The dominant type of messages in a mobile network are *Packet-in* and *Port-status* messages, and the exchange of the other messages is relatively small. This is the rationale behind modeling only these two types of messages.

Each controller is responsible for the forwarding decisions of all the switches in its domain. There is no hierarchical structure, meaning that a single controller has no info or access to all the switches, instead it exchanges information about the network with other controllers (synchronization and/or invocation messages). As the number of controllers increases, the total traffic handled per controller decreases, but the signaling between controllers becomes higher. When installing a new flow in the switch, if all the switches in the path of the packet belong to the same controller, a single controller installs all flow entries in all the switches. In the opposite case, a single controller can only install flow entries in the switches it controls and additionally must invoke other controllers to install the flows in their respective domains. Therefore, we propose a controller model with two different types of jobs served by the controller at different rates (single controller vs multiple controllers' case).

We assume that the controller and switches have infinite buffer size, and we model both as a single queue and not per interface. In the proposed model, we assume that the service rates of the switches and the controllers are load independent. For the sake of simplicity, the destinations of flows are uniformly distributed, and the packet arrivals can be modeled using a Poisson process. We use the conclusions from Kirsal and Gemikonakli [21], where the authors have shown that the handover calls distribution can be modeled as Poisson, meaning that the *Port-status* events occur following this distribution. The service discipline is First-In, First-Out (FIFO), and we consider having an open queueing network, where packets can enter and depart the system from any node. Additionally, we assume that the number of *off-port* messages is the same as the number of *on-port* messages (MN has to disconnect from one switch and to immediately connect to another switch, hence handover). This assumption is reasonable because the number of new MNs that "join" the network is significantly smaller compared to the number of active MNs that attempt a handover due to their mobility. Finally, we assume only TCP traffic, where only the header of the first packet is sent to the controller (*Packet-in* message).

3.3. Switch-to-single controller model

We model only *Packet-in* and *Port-status* messages. We analyze the case when MN disconnects from one switch, say s_i, and it attaches to another switch, s_j, where $i, j \in \{1, 2, \ldots, m\}$, and m is the number of switches in the domain. This triggers two *Port-status* (*off-port* and *on-port*) messages sent to the controller to update it during MN's mobility. When receiving the above messages, the controller sends *Flow-mod* to some of the switches to change the flow entries for the specific MN. The exchange of messages initiated by the mobility of the node is presented in Fig. 2, where "$^{(1)}$" denotes the *off-port*, and "$^{(2)}$" the *on-port* initiated message sequence.

If Λ_{is} is the total arrival rate of port modifications from the network to node i, then it can be modelled as:

$$\Lambda_{is} = \lambda_{is} + \sum_{j=1,\ j\neq i}^{m} \lambda_{js} \times a_{ij}, \tag{1}$$

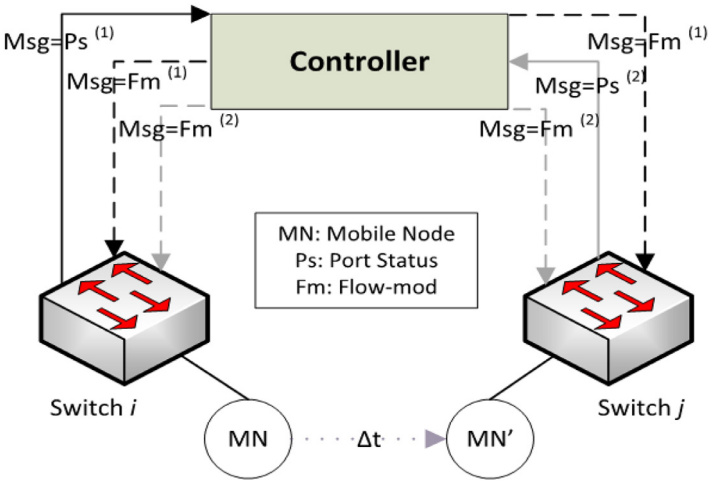

Fig. 2. Mobility triggered exchange of messages.

where λ_{is} and λ_{js} are the arrival rate of *Port-status* messages initiated by node i and j respectively, $a_{ij} = 1$ in case s_i flow table needs to be updated due to s_j *Port-status* message, otherwise $a_{ij} = 0$. For the controller, the total arrival rate of *Port-status* messages is:

$$\Lambda_{cs} = \sum_{i=1}^{m} \lambda_{is}. \tag{2}$$

The *Packet-in* messages are treated differently from the *Port-status* messages, so our queuing network contains two job classes, and we are considering a *multiclass* network. We define the general arrival rate of packets at node i as λ_{ip}, q_i^{pc} is the probability of a *Packet-in* message to be sent to the controller, c_{ij} is the routing probability between switches i and j, and $b_{ji} \in \{0,1\}$ is an indicator that describes whether the update from the node j involves node i in the flow path, or it does not. The total arrival rate of packets can be presented as:

$$\Lambda_{ip} = \lambda_{ip} + \sum_{j=1,\ j\neq i}^{m} \lambda_{jp} \times c_{ij} + q_i^{pc}\ \lambda_{ip} + \sum_{j=1,\ j\neq i}^{m} (q_j^{pc} \times b_{ji})\ \lambda_{jp}, \tag{3}$$

and the arrival rate of *Packet-in* messages to the controller is:

$$\Lambda_{cp} = \sum_{i=1}^{m} q_i^{pc}\ \lambda_{ip}. \tag{4}$$

We take the arrival rates defined in Eq. (1)–Eq. (4) and we substitute them in the following equations:

$$\rho_{is} = \frac{\Lambda_{is} + \Lambda_{ip}}{\mu_{is}}, \quad \rho_c = \frac{\Lambda_{cs} + \Lambda_{cp}}{\mu_c}, \tag{5}$$

where ρ_{is} and ρ_c are the load, and μ_{is} and μ_c are the mean service rate for the switch and the controller, respectively. Finally, we equate ρ_{is} and ρ_c at *load* ~ 1 and we calculate the values of λ_{is} and λ_{ip}. In practice, λ_{is} is expected to be much smaller than λ_{ip}.

3.4. Multiple controllers model

We consider a queueing model for the controller with aim to estimate the average sojourn time. To approximate the service time, we use Phase-Type (PH) distribution [22], commonly specified as a vector-matrix tuple (α,T), where:

$$\alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots.., \alpha_n),\ T = \begin{pmatrix} t_{11} & \cdots & t_{in} \\ \vdots & \ddots & \vdots \\ t_{n1} & \cdots & t_{nn} \end{pmatrix}. \tag{6}$$

The square matrix T has nonnegative off-diagonal elements and negative diagonal elements, α is the initial probability vector, and n is the number of transient states. The mean of PH distribution is given as:

$$E[X] = \alpha(-T)^{-1}e = \frac{1}{\mu_c}, \tag{7}$$

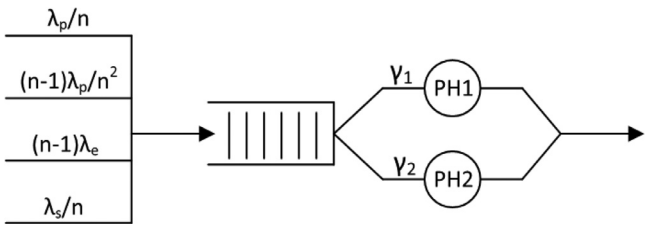

Fig. 3. The queue model at the controller.

where e is a column vector of ones with appropriate size, μ_c is the mean service rate of the controller. The server utilization is $\rho = = \lambda_c/\mu_c$, where λ_c is the total arrival rate at the controller.

The arrival rate of all *Packet-in* messages to the controllers is λ_p, with n controllers in the network, the *Packet-in* messages at each controller arrive at rate $\lambda_{cp} = \lambda_p/n$. The arrival rate of invocations is λ_{ci}, and for a single controller there are $(n-1)$ controllers sending invocation messages at rate of $\lambda_p/(n-1)$. Therefore:

$$\lambda_{ci} = \frac{(n-1)\lambda_p}{n^2}. \tag{8}$$

The arrival rate of synchronization messages at one controller is noted as λ_e, so the total arrival rate of message exchange λ_{ce} is given as:

$$\lambda_{ce} = (n-1)\lambda_e. \tag{9}$$

We denote with λ_{cs} the total arrival of *Port-status* messages, hence for a single controller:

$$\lambda_{cs} = \frac{\lambda_s}{n}. \tag{10}$$

Finally, the total arrival rate of messages at the controller can be given with the following equation:

$$\lambda_c = \lambda_{ce} + \lambda_{ci} + \lambda_{cp} + \lambda_{cs}. \tag{11}$$

The queueing system is presented in Fig. 3.

If the controller can handle the request alone, then it takes time $t_1 \sim PH(\alpha_1, T_1)$, otherwise it takes time $t_2 \sim PH(\alpha_2, T_2)$. We define γ_1 as the probability that a controller spends time t_1 on a job, and $\gamma_2 = 1 - \gamma_1$ is the probability the controller spends time t_2 on a job, hence:

$$\gamma_1 = \frac{n^2(n-1)\lambda_e + n(\lambda_p + \lambda_s)}{n^2(n-1)\lambda_e + n(\lambda_p + \lambda_s) + (n-1)(\lambda_p + \lambda_s)}. \tag{12}$$

We denote the service time with $PH(\alpha,T)$. The distribution of the controller's service time is a combination of $PH(\alpha_1,T_1)$ and $PH(\alpha_2,T_2)$, hence:

$$\alpha = (\gamma_1\alpha_1, \gamma_2\alpha_2), \quad T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}. \tag{13}$$

An M/PH/1 queue can be studied as a Quasi-Birth-and-Death (QBD) process with the state space $E=\{0, (i, j), i \geq 1, 1 \leq j \leq v\}$, where v is the number of phases. The state 0 presents the case of empty queue, the state (i, j) of having i customers in the system, while the service process is in the phase j. The generator Q of that Markov process is the following:

$$Q = \begin{pmatrix} -\lambda & \lambda\alpha & 0 & 0 & 0 \\ \tau & T-\lambda I & \lambda I & 0 & 0 \\ 0 & \tau\alpha & T-\lambda I & \lambda I & 0 \\ 0 & 0 & \tau\alpha & T-\lambda I & \lambda I \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix}, \tag{14}$$

where I is a square matrix with ones as diagonal elements, and $Te = -\tau$. The generator Q has a stationary probability vector given as $x = (x_0, x_1, x_2, \ldots)$.

The steady state equations are given by Neuts [23] in Theorem 1 as:

$$-\lambda x_0 + x_1\tau = 0, \tag{15}$$

$$\lambda x_0\alpha + x_1(T-\lambda I) + x_2\tau\alpha = 0, \tag{16}$$

$$\lambda x_{i-1} + x_i(T-\lambda I) + x_{i+1}\tau\alpha = 0, i \geq 2. \tag{17}$$

If we multiply Eqs. (16) and (17) by the column vector e on the right, we obtain:

$$x_{i+1}\ \tau = \lambda x_i e, i \geq 1. \tag{18}$$

Multiplying Eq. (18) by α on the right and combing with Eq. (17), we get:

$$x_i\ (\lambda I - \lambda e\alpha - T) = \lambda x_{i-1},\ \ i \geq 2, \tag{19}$$

$$x_1(\lambda I - \lambda e\alpha - T) = \lambda x_0 \alpha. \tag{20}$$

Therefore:

$$x_i = x_0 \alpha R^i, \tag{21}$$

$$R = \lambda(\lambda I - \lambda e\alpha - T)^{-1}, \tag{22}$$

where R is a matrix as defined in (22). Since x is the stationary probability vector,

$$\sum_{i=0}^{\infty} x_i = 1, \tag{23}$$

the following is obtained:

$$\begin{aligned}
\sum_{i=0}^{\infty} x_i = x_0 + x_0\alpha \sum_{i=1}^{\infty} R^i e &= x_0 + x_0 \alpha R(I-R)^{-1} e \\
&= x_0 - \lambda x_0 \alpha (\lambda e\alpha + T)^{-1} e \\
&= x_0 - \lambda x_0 \alpha T^{-1}\left(I + \lambda e\alpha T^{-1}\right)^{-1} e \\
&= x_0 - \lambda x_0 \alpha T^{-1} \sum_{k=0}^{\infty} (-1)^k \lambda^k \left(e\alpha T^{-1}\right)^k e \\
&= x_0 - \lambda x_0 \alpha T^{-1}\left(I - \lambda(1-\rho)^{-1} e\alpha T^{-1}\right) e \\
&= x_0 + x_0\rho + x_0\rho^2(1-\rho)^{-1},
\end{aligned} \tag{24}$$

so that $x_0 = 1 - \rho$.

The average number of jobs in the queue can be obtained using:

$$\begin{aligned}
E(k_c) = \sum_{i=1}^{\infty} i x_i e &= \sum_{i=1}^{\infty} i x_1 R^{i-1} e \\
&= x_1 \sum_{i=1}^{\infty} \frac{d}{dR} R^i e = x_1 \frac{d}{dR}\left(\sum_{i=1}^{\infty} R^i\right) e \\
&= x_1 \frac{d}{dR}\left((I-R)^{-1} - I\right) e = x_1 (I-R)^{-2} e
\end{aligned} \tag{25}$$

Finally, the average flow setup time $E(t_c)$ is given as:

$$E(t_c) = \frac{E(k_c)}{\lambda_c} = \frac{x_1 (I-R)^{-2} e}{\lambda_c}. \tag{26}$$

3.5. Performance metric

Our aim is to quantify the handover delay which is a product of exchanging OF signaling messages in SDN-based mobile networks. For a *Port-status* event, we denote the mean interaction delay between switch i and the controller c as:

$$E\left[T_i^c\right] = E\left[t_i^s\right] + E\left[t_{i\_prop}^c\right] + E\left[t_i^c\right] + E\left[t_{i\_prop}^{c\_max}\right] \tag{27}$$

where t_i^s and t_i^c is the respective processing delay at the switch i and the controller in the domain, $t_{i\_prop}^c$ is the propagation delay of the *Port-status* message sent from the switch i to the controller, and $t_{i\_prop}^{c\_max}$ is the maximum propagation delay (required to reach the furthest switch) for the *Flow-mod* message sent to the subset of switches as a response of the controller to the *Port-status* message. The complete handover delay T_{ho} that the MN experiences is a sum of the delays introduced by both *off-port* message at switch i and the *on-port* message sent by switch j:

$$E[T_{ho}] = E\left[t_i^c\right] + E\left[T_j^c\right], \tag{28}$$

$$E[T_{ho}] = E\left[t_i^s\right] + E\left[t_j^s\right] + E\left[t_i^c\right] + E\left[t_j^c\right] + E\left[t_{prop}^{tot}\right], \tag{29}$$

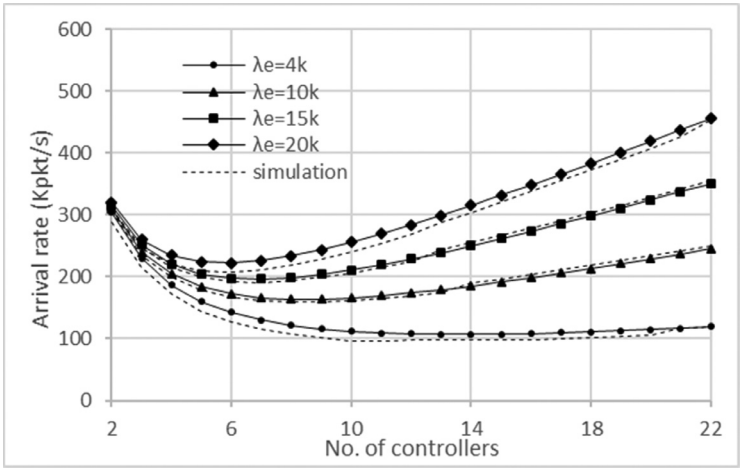

Fig. 4. Controller arrival rate.

where:

$$E\left[t_{prop}^{tot}\right] = E\left[t_{i\_prop}^{c}\right] + E\left[t_{i\_prop}^{c\_max}\right] + E\left[t_{j\_prop}^{c}\right] + E\left[t_{j\_prop}^{c\_max}\right]. \tag{30}$$

In practice, the processing delay of the switch is noticeably smaller than the processing delay of the controller [8]. The propagation time of the messages should be considered for mobile networks due to the large distances, however the handover delay is mainly contributed by the controller's service time as it will be shown in the next section.

4. Performance evaluation

In this section, we validate the proposed analytical model, by comparing it with the simulation results obtained with our discrete event simulator, developed in MATLAB. We use the latest version of MATLAB R2019a, and we run our simulations on a laptop with Intel Core i5-8350U 1.9 GHz with 16GB RAM. Our basic simulation scenario consists of 2 controllers and 10 switches per controller domain (we can easily add/remove controller/switches). There are 10 users connected to a switch, and the total arrival rate at each switch equals total system throughput divided by the number of switches. We can control the amount of traffic sent to the controllers (hence we can vary q^{pc}). During the simulation, all the switch-to-controller generated traffic is stored in files that are post-processed in MATLAB. We run each simulation 20 times (for 5 s each), and in the post-processing phase we calculate the mean value for each point in the graphs. The dominating messages at controller initialization phase are OFPT_FEATURES_REQUEST (REPLY) and OFPT_FLOW_MOD. For the controller packet sojourn time, we measure the time difference between receiving OFPT_PORT_STATUS and sending OFPT_FLOW_MOD message. We try to model a realistic small mobile operator with total system throughput of 5–20 Gbps. Based on a real mobile network with similar network throughput, we take the total network attach requests to be 40 req/s. For q^{pc} we use 4%, μ_{is} = 9.8 μs, μ_c = 240 μs [7], and the size of *Packet-in, Port-status and Flow-mod* messages is set at 128 B. The distance between the switches and the controller is 50–200 km, and the link between the switches and the controller is optical, 1 Gbps. We take a realistic number of switches per domain, m = 10. For the numerical results, we use the measurements in [4] and fit a Hyper-Erlang distribution to the controller's response time.

In Fig. 4 we analyze a scenario with variable number of controllers and its impact on the total arrival rate per single controller. The controllers exchange synchronization messages in steps, at total rate of 4–20k per second, and λ_p is fixed at 400k per second. We can see that the arrival rate drops when the number of controllers increases up to a certain point (e.g. for λ_e = 20k, up to n = =6), but after that point the arrival rate increases. When the controller number is low, the controllers can share a large number of *Packet-in* messages, and this will trigger only a few synchronization messages. In the opposite case, each controller handles smaller number of *Packet-in* messages, but this implies higher synchronization messaging. We also notice that for higher values of λ_e, the arrival rates are also higher as expected. Finally, we can draw a conclusion that for lower λ_e, in order to achieve a minimal arrival rate, the number of controllers needs to be higher compared to the case of achieving the same with higher values of λ_e.

Fig. 5 presents the controller's sojourn time in scenarios involving different number of controllers and with varying rate of *Packet-in* messages. The synchronization messages are fixed at 20k per second, while the arrival rate of *Packet-in* messages changes in the range of 100–600k per second. The flow setup time decreases as the controller number increases, but only up to n = =2 which is the inflection point beyond which the setup time increases almost linearly until n = 7. The conclusion is that even if more controllers are added to handle more *Packet-in* messages, still the extra controllers will impact the flow setup time due to the communication overhead. We also need to mention the severe degradation of the packet sojourn time in the case of a single controller, analyzed further in this section. Furthermore, we notice that after n = 7, the flow setup

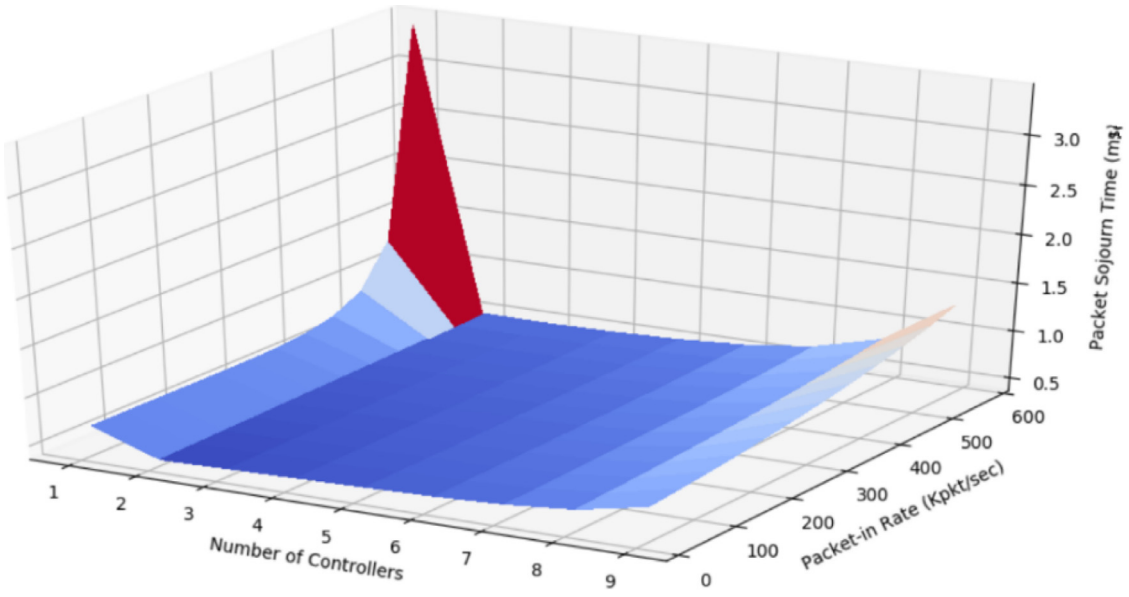

Fig. 5. Packet sojourn time (packet-in rate, no. of controllers).

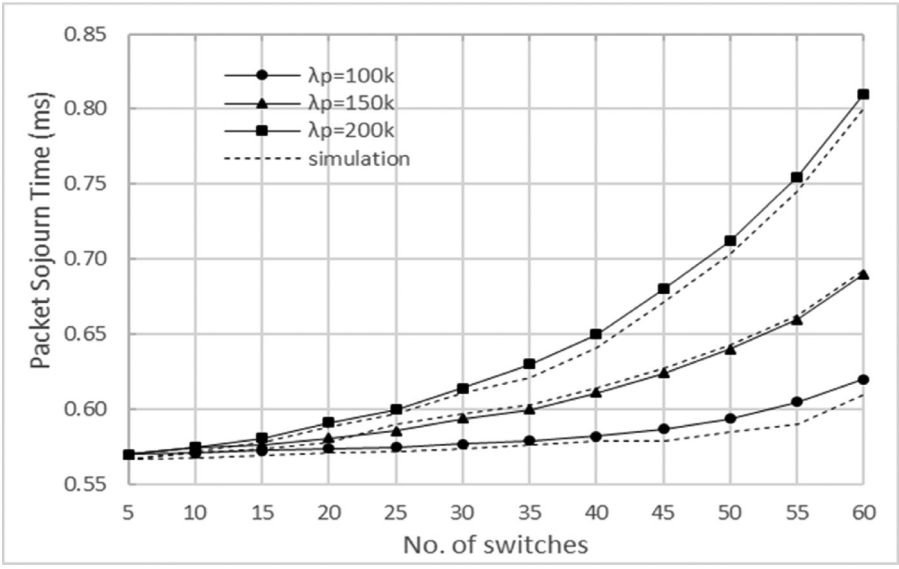

Fig. 6. Packet sojourn time (number of switches).

time rapidly degrades, as this is the threshold beyond which the number of synchronization messages becomes significantly high that causes degradation in performance.

In Fig. 6 we analyze the dependency of controller's service time on the number of switches by changing the arrival rate of *Packet-in* messages. As the number of switches increases, the controller's service time increases. However, it is noticeable that the packet sojourn time increases gradually, as the number of switches increases. When *Packet-in* messages arrive at the rate of 100k per second, the service time of a controller connecting to 50 switches is only about 0.04 ms higher compared to the case of connecting to a single switch. Additionally, we notice that the packet sojourn time rapidly degrades as the probability of the *Packet-in* messages increases. Our simulation results follow closely the analytical model, except for the case of $\lambda_p = 100$ k (when the number of switches is above 50).

Fig. 7 shows the packet sojourn time as a function of the controller load. As expected, when the load approaches 1, the delay approaches infinity, and as q^{pc} increases, the service time proportionally increases. We also notice that the analytical model follows the simulation closely for lower controller load values, and for lower values of q^{pc}, whereas for higher value of q^{pc}, the analytical model becomes less accurate. In [7], Jarschel et al. show that the probability of new flows is 4%, so

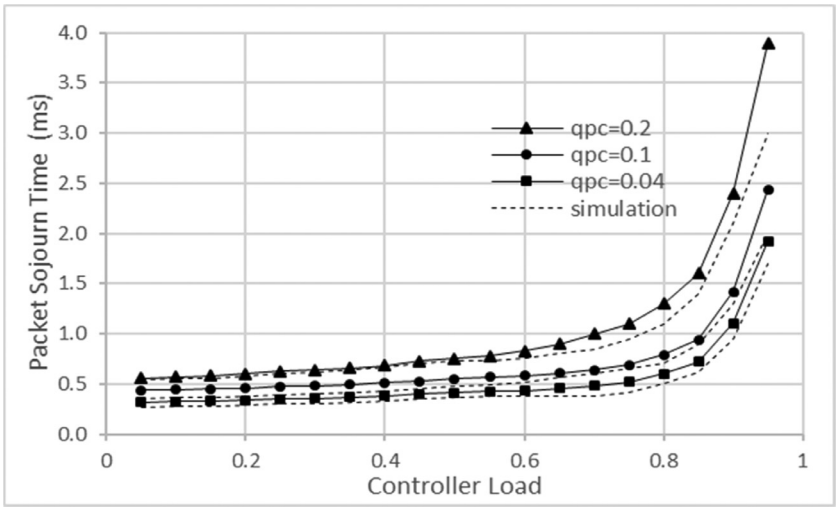

Fig. 7. Packet sojourn time (load), effect of q^{pc}.

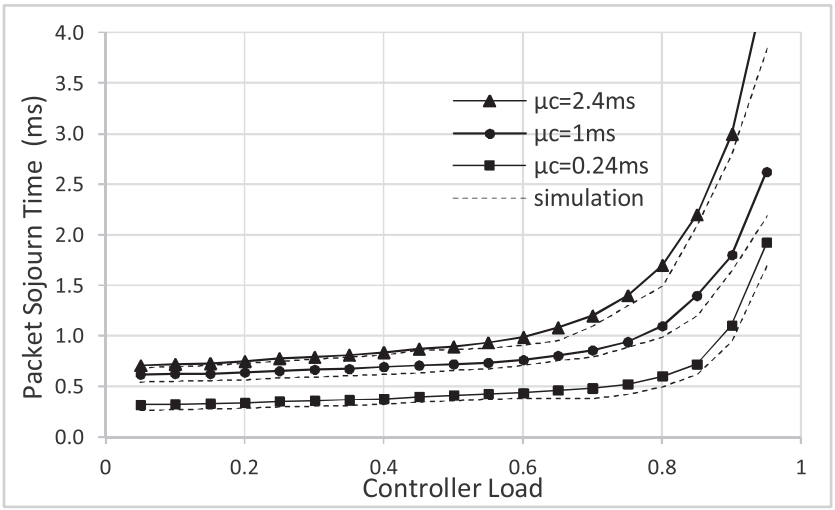

Fig. 8. Packet Sojourn time (load), effect of μ_c for a fixed value of q^{pc}.

we can conclude that the proposed model is safe to be used. We also want to investigate the impact of the controller's mean service time on the packet sojourn time, while keeping $q^{pc} = 0.04$. The results are shown in Fig. 8. As expected, the packet sojourn time decreases for higher values of μ_c. This plot can be used for setting guarantees on the packet sojourn time during the network design, where the controller has a fixed mean service rate and a limited load. For a controller load around 0.7 to 0.8, the packet sojourn time sharply increases, and becomes unacceptably high for the higher range of load values.

By applying the particle swarm optimization algorithm [4] on Eq. (26), we try to get the optimal number of controllers to achieve minimal flow setup time while varying the synchronization rate and the *Packet-in* rate. The results in Fig. 9 show that, as the synchronization rate increases, the optimal number of controllers decreases. For the *Packet-In* messages, the effect is opposite, as λ_p increases, the number of controllers needed to achieve minimum setup time also increases. If there are many controllers in the network, the synchronization messages cause a high overhead, and this impacts the flow setup time, whereas higher number of controllers are needed to handle effectively a large number of switches/users. The results obtained from Fig. 9 are in-line with those in Fig 5, where it can be easily noticed that for $\lambda_p = 20$k, the optimal number of controllers is 2. Going back to Fig. 5, we can also conclude that in the case of a single controller, the packet sojourn time is severely degraded. This would mean that a SDN network designer should never deploy a single controller for a small network operator. To further inspect this phenomenon, we made additional analytical model analysis and simulations with very high synchronization rates ($>$100k). Our findings reveal that only in this particular case of extremely high overhead, a single controller may be the optimal choice, however it must be emphasized that this scenario is highly unlikely.

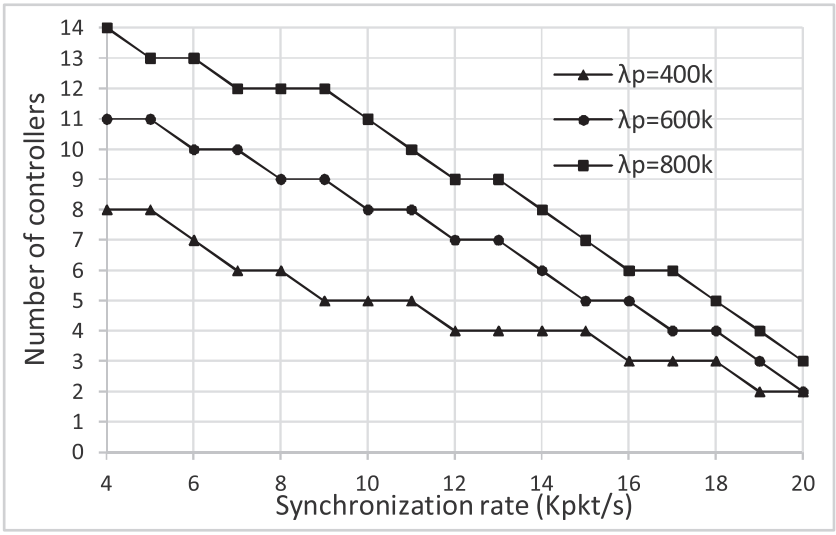

Fig. 9. The optimal number of controllers.

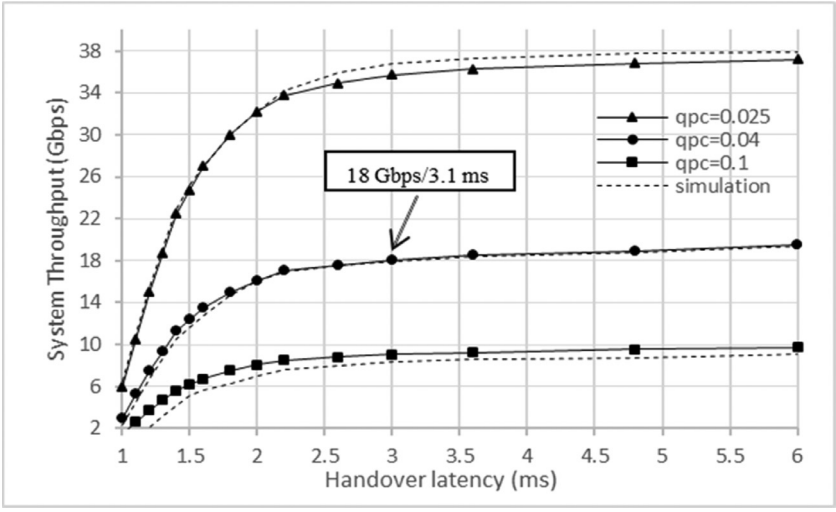

Fig. 10. System throughput (handover latency).

Finally, Fig. 10 shows the targeted system throughput as a function of the total handover delay that the MN experiences during a "hard" handover. System throughput is the total traffic handled by the network. We assume number of controllers $n = =2$, the synchronization messages are at fixed rate of 20k per second, and q^{pc} has three different values, hence three curves. The plot shows that initially the handover latency moderately increases, but then reaches a network saturation point. Beyond this point, the handover latency sharply increases, whereas the throughput negligibly increases. This analysis shows the elasticity limit of the network and can be useful for designing the sojourn time guarantees of a mobile network for a given amount of traffic that is directed towards the controller. For example, for $q^{pc} = =0.04$ and network throughput of 18 Gbps, the handover delay due to OF signaling is 3.1 ms. Additionally, we notice that when the probability of *Packet-in* messages is high, the handover latency sharply increases even for small values of the system throughput. In other words, the designers must work on lowering the expected q^{pc} in order to provide higher QoS to the customer by keeping the handover delay at acceptable value.

5. Impact of model's assumptions evaluation

In order to obtain closed-form equations, we made several major model assumptions in Section 3.2, such as: unlimited buffer, modeling the switch/controller as a single queue, Poisson arrival rate, and switch/controller's service rates are load-independent. In this section, we discuss the impact of each of the limitations on the model accuracy, and we perform additional simulations to quantify this impact.

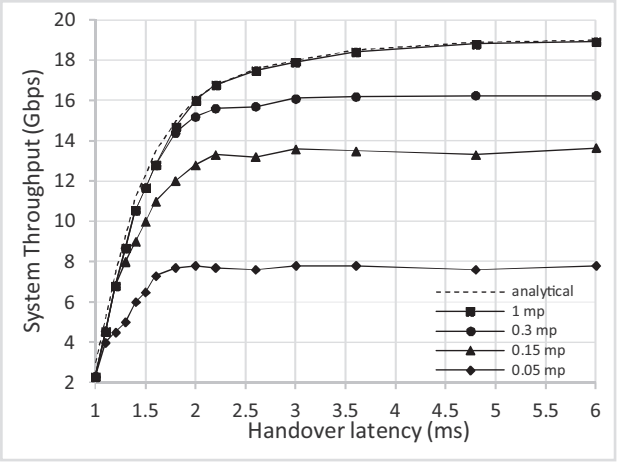

Fig. 11. Impact of the limited controller buffer on the handover latency.

In a real OF-switch, there are multiple ingress and output ports, and every ingress port has a fix buffer size. The incoming packets from each buffer are processed against the same flow tables, and there is no control traffic prioritization (each packet is processed according to the FIFO principle). It must be emphasized that some OF-switches incorporate traffic priority, but such implementations are not subject of our analysis. In our initial simulations, for both the controller and switches, we have set the buffer size to a very high value (1 million packets (mp)), however this time we vary the total buffer size in the range of 0.05 mp to 1 mp. Additionally, we set multiple number of queues (2 and 6) for the switch, and the arrival rate per switch varies in the range of 100–500 Kpkt/s. We set $q^{pc} = 0.04$, our packet size remains at 128B, and we run our simulations 20 times with simulation duration of 5 s.

The first conclusion we obtain from the simulation results is that for a fixed buffer size per switch (and fixed mean service rate), the maximum packet arrival rate before any packet loss (due to insufficient buffer space) remains the same. This implies that even with the increase of the number of buffers (interfaces), the maximum packet rate before any packet loss per buffer decreases, and on switch level the rate remains constant. We notice that as the arrival rate increases, the minimal buffer size to ensure no packet loss also needs to increase as expected. In our analytical model, we have justifiably assumed unlimited buffer for the switch, as a very large buffer is typically found in hardware implementations of OF-switches. Additionally, the switch processing time contributes slightly in the total handover latency, as the dominating factor is the controller processing time, so even a severe performance degradation due to small buffer size of the switch will not change noticeably the value of the total handover delay.

In Fig. 11, we investigate the impact of the limited buffer size at the controller, and show the simulation results of the system throughput dependency from the handover latency. We aim to evaluate the accuracy of Fig. 10, where we used a sufficiently large buffer to avoid any packet loss. We take the same simulation parameters, keep $q^{pc} = 0.04$, and vary the buffer size for three finite values of 0.05, 0.15 and 0.3 mp. A severe performance degradation is noticeable for buffer size of 0.05, whereas the handover delay difference from the analytical model for buffer size of 0.3 mp is small. Due to the packet loss and the TCP congestion avoidance mechanisms, the handover latency drastically increases as the buffer size decreases. We come to similar conclusions and we see the same effect of limiting the buffer size for the results shown in Figs. 4 and 6–8. As the number of *Packet-In* messages at the controller increases, for smaller buffer sizes of the controller, the degradation of the packet sojourn time is more tangible. We conclude that to improve the accuracy of our proposed model, in our next work we should consider the effect of limiting the buffer size at the controller, whereas the effect of modeling the OF-switch and controller as a single buffer is not an impactful limitation when traffic prioritization is not used.

In the analytical models proposed to study the OF networks, the most frequent assumption considered is the Poisson arrival rate. In most of the references that we analyzed, this assumption was applied, however we know for a fact that the real network traffic exhibits self-similar characteristics. The self-similar traffic is a widely-spread traffic phenomenon in modern networks, and its basic characteristic is: segments of the process have the very same statistical properties at different scales. The important definitions and properties of self-similarity are given by Leland et al. [24], where the authors define an important parameter to quantify the degree of self-similarity, called the Hurst parameter. The simulation of self-similarity has also attracted the attention from the research community, e.g. in [25] Tomic and Maletic compare several traffic generation models implemented in MATLAB. In our future work, we plan to investigate the effect of traffic self-similarity as our theoretical analysis shows that the effect of this phenomenon can have an impact on evaluating the performance of SDN-based mobile networks.

Finally, the assumption in the proposed model for load independency of the mean service rate has a more predictable effect on the modeling accuracy. The average processing time in the real implementations of OF-switch and OF-controller

tends to deteriorate at higher loads. We would expect that this should affect results in Figs. 7 and 8, where we foresee that the performance degradation will be much more visible at loads > 0.8. However, the effect of load dependent service rate can be effectively mitigated with careful SDN design considerations, by selecting a controller with sufficiently high mean service rate.

6. Conclusion

In this work, we analytically model the delay introduced when the Mobile Node performs a handover and moves from one switch to another in SDN-based mobile core networks. We use queuing theory to analyze the controller packet sojourn time and aim to quantify the network throughput as a function of the total handover delay. We also confirm that packet arrival rate, controller load, and number of network elements are critical factors that require considerations for modeling OpenFlow networks. We demonstrate that a high probability of *Packet-In* messages, and a high number of controllers (when many synchronization messages are exchanged), have a negative impact on the packet service time and degrade the network performance. Additionally, an analysis is performed for estimating the optimal number of controllers for a fixed *Packet-In* and synchronization rates. The analytical model is validated via simulations. The obtained results provide valuable conclusions that can facilitate the design of OpenFlow-based networks. As future work, we plan to incorporate hardware considerations into the model and compare the results from the current proposal with real hardware implementations. We also aim to setup a multi-controller test-bed to experimentally evaluate the accuracy of the modeling, and plan to use synthetic traffic generators that would mimic the self-similar nature of the internet traffic. Finally, we will introduce a mathematical approach for quantifying the packet sojourn time of the switches in order to increase the accuracy of our handover delay model.

Declaration of Competing Interest

None.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2019.106546.

CRediT authorship contribution statement

Strahil Panev: Conceptualization, Methodology, Software, Writing - original draft, Visualization, Investigation. **Pero Latkoski:** Supervision, Methodology, Resources, Validation, Writing - review & editing.

References

[1] Cox JH, Chung J, Donovan S, Ivey J, Clark RJ, Riley G, et al. Advancing software-defined networks: a survey. IEEE Access 2017;5:25487–526.
[2] Open Networking F.. OpenFlow switch specification version 1.3.1. Tech. Republic; 2012.
[3] Issa T, Raoul Z, Konaté A, Adepo JC, Cousin B, Olivier A. Analytical load balancing model in distributed open flow controller system. Sci Res Eng 2018;10(12):863–75.
[4] Zhihao S, Wu H, Wolter K. Performance evaluation of the control plane in software defined networks. the 12th EAI International conference; 2019.
[5] Mahmood K, Chilwan A, Osterbo O, Jarschel M. Modelling of openflow-based software-defined networks: the multiple node case. IET Netw 2015;4(5):278–84.
[6] Sarkar C, Setua SK. Analytical model for openflow-based software-defined network. In: Progress in computer, analytics and networking; 2018. p. 583–92.
[7] Jarschel M, Oechsner S, Schlosser D, Pries R, Goll S, Tran-Gia P. Modeling and performance evaluation of an openflow architecture. In: 23rd International teletraffic congress (ITC); 2011. p. 1–7.
[8] Thieme C. Challenges for modelling of software-based packet processing in commodity hardware using queueing theory. Netw Arch Serv 2017;49.
[9] Mahmood K, Chilwan A, Østerbø O, Jarschel M. On the modeling of openflow-based SDNs: the single node case. Comput Sci Inf Technol 2014;4:207–14.
[10] Shang Z, Wolter K. Delay evaluation of openflow network based on queueing model. 12th European dependable computing conference (EDCC 2016); 2016.
[11] Blial O, Mamoun MB, Benaini R. An overview on SDN architectures with multiple controllers. J Comput Netw Commun 2016;2016.
[12] Sood K, Yu S, Xiang Y, Cheng H. A general QoS aware flow-balancing and resource management scheme in distributed software-defined networks. IEEE Access 2016;4:7176–85.
[13] Chen X, Zhao B, Ma S, Chen C, Hu D, Zhou W, et al. Leveraging master-slave openflow controller arrangement to improve control plane resiliency in SD-EONs. Opt Express 2015;23(6):7550–8.
[14] Heller B, Sherwood R, McKeown N. The controller placement problem. In: Proceedings of the first workshop on Hot topics in software defined networks. ACM; 2012. p. 7–12.
[15] Hu Y, Wendong W, Gong X, Que X, Shiduan C. Reliability-aware controller placement for software-defined networks. In: 2013 IFIP/IEEE International symposium on integrated network management (IM 2013). IEEE; 2013. p. 672–5.
[16] Marquezan C, An X, Despotovic Z, Khalili R, Hecker A. Identifying latency factors in SDN-based mobile core networks. In: 2016 IEEE Symposium on computers and communication (ISCC); 2016. p. 484–91.
[17] Duan X, Akhtar AM, Wang X. Software-defined networking-based resource management: data offloading with load balancing in 5g hetnet. EURASIP J Wirel Commun Netw 2015;181(1) 2015.
[18] Meneses F, Corujo D, Guimaraes C, Aguiar A. Extending SDN to end nodes towards heterogeneous wireless mobility. 2015 IEEE Globecom workshops (GC Wkshps); 2015.

[19] Alotaibi M, Helmy A, Nayak A. Modeling handover signaling messages in openflow-based mobile software-defined network. J Comput Netw Commun 2018;2018.
[20] Panev S, Latkoski P. SDN-based failure detection and recovery mechanism for 5G core networks. Transactions on emerging telecommunication technologies. Wiley; 2019.
[21] Kirsal Y, Gemikonakli O. Performability modelling of handoff in wireless cellular networks with channel failures and recovery. In: 2009 11th International conference on computer modelling and simulation; 2009. p. 544–7.
[22] Gillent F, Latouche G. Semi-explicit solutions for M/PH/1-like queuing systems. Eur J Oper Res 1983;13(2):151–60.
[23] Neuts MF. Explicit steady-state solutions to some elementary queueing models. Oper Res 1982;30(3):480–9.
[24] Leland WE, Taqqu MS, Willinger W, Wilson DV. On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Trans Netw 1994;2(1):1–15.
[25] Tomic I, Maletic N. Comparison of models for self-similar network traffic generation. In: X International symposium on industrial electronics (INDEL); 2014. p. 266–9.

Strahil Panev, received his M.S. degree in telecommunications in 2011 at the Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University in Skopje. He is currently pursuing the Ph.D. degree at the same university. His research interests include SDN, mobile networks, and analytical modeling.

Pero Latkoski, received his M.S. and Ph.D. degree at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University in Skopje, in 2006 and 2010, respectively. He currently holds the position of full professor at the Institute of Telecommunications, at the same university. His research interests include communication protocol engineering, software defined networking, and information theory.