Performance Analysis of General P4 Forwarding Devices with Controller Feedback: Single- and Multi-Data Plane Cases[⋆,⋆⋆]

Nicolai Kröger[a,∗], Fidan Mehmeti[a], Hasanin Harkous[b], Wolfgang Kellerer[a]

[a] *Technical University of Munich, Arcisstraße 21, Munich, 80333, Bavaria, Germany*
[b] *Nokia, Werinherstraße 91, Munich, 81541, Bavaria, Germany*

Abstract

Software-Defined Networking (SDN) lays the foundation for the operation of future networking applications. The separation of the control plane from the programmable data plane increases the flexibility in network operation, and hence, improves the overall performance. One of the most used languages for describing the packet behavior in the data plane is P4. It allows both protocol and hardware independent programming. With the expanding deployment of P4 programmable devices, it is of utmost importance to understand their achievable performance and limitations in order to design a network and provide Quality of Service (QoS) guarantees in terms of different metrics of interest to users communicating in the network. One of the most important figure of merits is the mean sojourn time of a packet in a P4 device. While previous works already modeled the sojourn time in P4 devices with controller feedback, those models were rather oversimplified and could not capture the real system behavior for general cases, resulting this way in a potentially high inaccuracy in performance prediction. To bridge

[⋆]Initial results for this work have been presented at the 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) 2022 [1].

[⋆⋆]Funding: This work was supported by the Federal Ministry of Education and Research of Germany (BMBF) under the project "6G-Life", with project identification number 16KISK002. The funding source had no other involvement with the work.

[∗]Corresponding author.

Email addresses: `nicolai.kroeger@tum.de` (Nicolai Kröger), `fidan.mehmeti@tum.de` (Fidan Mehmeti), `hasanin.harkous@nokia.de` (Hasanin Harkous), `wolfgang.kellerer@tum.de` (Wolfgang Kellerer)

this gap, in this paper, we consider the system behavior of P4 devices for the general case, i.e., under general input parameter distributions. To that end, we model the system behavior with a queueing network with feedback. First, we do this for a single data plane, and then we extend the analysis to the case when there are multiple data planes sending occasional packets to the same controller. Due to the fact that it is impossible to provide closed-form solutions in the general case, we consider different approximation approaches for the mean sojourn time and show which one is better for a given scenario. We validate our results against extensive realistic simulations, capturing different behaviors in the data and control planes. Results show that the most accurate approximation in most cases is the one in which queueing networks are decoupled and considered as independent queues despite the fact that there are considerable dependencies involved. The level of discrepancy with the best approximating approach in the worst case for a single data plane does not exceed 18.2% for service times distributions with a coefficient of variation not greater than 1, whereas when dealing with multiple data planes, the discrepancy is usually higher, but with the best-approximating approach in each case rarely exceeds 14%.

Keywords: P4, SDN, Queueing networks with feedback, Diffusion approximation, M/G/1.

1. Introduction

Emerging applications in the contemporary digitized world, such as telemedicine or autonomous driving, impose stringent requirements on the underlying communication networks in terms of latency, throughput, reliability, and more. In order to satisfy these requirements, Software-Defined Networking (SDN) [2] emerged as a new networking paradigm which decouples the data plane from the control plane. This approach allows optimizing the network management in a holistic way as one controller can define the packet processing in several forwarding devices. Furthermore, the introduction of Programmable Data Planes (PDP) [3] enables controlling the packet processing with greater granularity. One of the most well-known concepts for data plane programming is the domain-specific P4 language concept [4], which is used for defining the packet processing in the forwarding devices in detail.

As P4 devices are becoming more and more popular in networking, it is of utmost importance to understand their performance behavior and limitations. Having this information before their deployment enables designing properly a network with regards to Quality of Service (QoS) guarantees such as low latency, high throughput, or high reliability. One of the most important performance metrics is the mean packet sojourn time, i.e., the mean time a packets spends in the system. In a first step to model the packet sojourn time for P4 devices, the authors in [5] measured the mean sojourn time of packets for the P4-enabled devices Netronome SmartNIC [6], NetFPGA [7], and T4P4S [8], based on the complexity of the loaded P4 program. Those measurements were then further used in [9], where one P4 forwarding device with controller feedback is modeled as a Jackson network [10]. Their results allow a first understanding of the behavior and limitations of a P4 device with controller interaction. However, as service times in [9] are assumed to be exponentially distributed, both in the data plane and the controller, it may not reflect the behavior of real devices and therefore the analytical performance results can vary from the real ones considerably.

Refining the system model with more general and hence, realistic distributed services times is not a straightforward task though. Relaxing the assumption of exponentially distributed service times leads to a general system where the well-known exact equations for the packet mean sojourn time in the system do not hold anymore. Moreover, the real behavior can only be approximated as it is not possible to obtain closed-form exact analytical expressions. Another problem lies in obtaining measurement results for all possible combinations of real data planes and controllers to compare with the analytical results; this is cumbersome. This is even more emphasized when there are multiple data planes served by a controller in the network. Hence, finding a way to analyze approximation techniques with results close to reality would significantly improve the understanding related to the performance of P4 devices.

In particular, several important research questions arise in the context of the performance of P4 devices, and more specifically, the packet mean sojourn time in a P4-enabled network:

- How can the packet mean sojourn time in realistic P4 devices be obtained analytically? How well does the model fit to the actual results? How does the distribution of service times affect the accuracy of the model?

- What are the limitations of such models? Are they more vulnerable when considering multiple data planes?

- How does the interaction with the controller impact the packet mean sojourn time? Does loading the controller with more requests impact the accuracy of the model?

These questions are addressed in this paper by modeling a P4 forwarding device with controller feedback using a queueing theoretic model first. We assume both low-variance (Erlang), exponential, and high-variance (hyper-exponential) distributed service times on the data plane and control plane. Further, we analyze three approximation approaches for the packet mean sojourn time in such systems and compare the analytical results to those obtained by simulation in order to cover a broad operation region, by varying a wide amount of parameters. Then, we extend the analysis to multiple data planes, i.e., multiple switches that a packet traverses, and a single controller. For this case as well, we consider different distributions for the time packets spend in data planes and the controller and provide different approximations while showing which one is more suitable in a given scenario. We provide extensive evaluation of the performance, where the results enable to observe the accuracy of the approximation approaches in terms of the packet mean sojourn time of P4 devices and therefore to understand the behavior of our metric of interest as a function of different parameters. The main message of this work is for most scenarios of interest the most accurate approximation is the one in which the queues are decoupled and are considered as mutually independent M/G/1 queues.

Specifically, our main contributions are:

- We analyze different approximation methods for the mean sojourn time of a packet in a P4 forwarding device with controller feedback and generally distributed service times.
- We compare the analytical results obtained by the approximation approaches to those obtained by simulation for widely varied parameters and operation regions.
- We also consider the scenario in which there are multiple P4 forwarding devices and a controller and provide the approximation results for general service times across all entities.

The remainder of this paper is organized as follows. In Section 2, some related work is discussed. Then, in Section 3 some background information on the P4 programming language is provided. Further, in Section 4 the queueing model of a P4 forwarding device with controller feedback is presented. This is followed by a theoretical analysis of approximation techniques for the packet mean sojourn time of such a system in Section 5. The model and the analysis for multiple connected P4 forwarding devices are given in Section 6. The performance for a vast range of scenarios is evaluated in Section 7. Finally, Section 8 concludes the paper.

2. Related Work

The introduction of SDN enabled splitting the control plane from the data plane. Since then, many works in the literature targeted modeling the performance of these devices, but most of them making simplifying assumptions for analytical tractability.

OpenFlow (OF)-based switches were modeled in [11] as a feedback-oriented queueing system similar to that studied in this paper. However, in [11] each queueing system at the control and data plane is assumed to have exponentially distributed service times. A follow up work in [12] refactors the model and presents it as a Jackson network. A model for networks made up of such subsystems is proposed and analyzed in [13]. Goto *et al.* [14] improves on the model presented in [11] for a single node by incorporating processing priorities for packets going to the switch. Ansell *et al.* [15] introduce a control plane application based on queueing theory for monitoring networks and predicting their behavior.

P4 programmable data planes extended the programmability offered by the SDN paradigm to the data plane. Similar to the approach presented in [11], the model in [9] adopts a feedback-oriented queueing system to abstract the behavior of the system. Moreover, it assumes that the processing delay, and thus the forwarding delay, at the data plane can vary based on the complexity of the loaded P4 data path. Accordingly, the average service time of the data plane is modeled using an exponential distribution based on the P4 pipeline's complexity. The impact of different P4 atomic operations on the processing delay of different P4 devices is evaluated in [5], where a method is proposed to estimate the packet forwarding delay on different P4 devices as a function of the complexity of the loaded P4 program.

Other works in the literature evaluate and model the performance of P4 programmable devices using different approaches. For example, a benchmarking suite for P4 programmable devices is proposed in [16] for evaluating the performance of three different P4 devices. The authors in [17] evaluate and model the performance of hardware- and software-based P4 devices according to different criteria considered relevant to each type of device. They model the packet rate that could be handled by the T4P4S software switch when the number of incoming flows increases. On the other hand, the authors in [17] decided to focus on modeling the usage of processing resources, i.e., the utilization ratio, on TOFINO ASIC switches as they identified this metric to be more important for this type of devices. In another work, Helm *et al.* [18] propose a network calculus-based model for the worst case sojourn time in the data plane without controller feedback.

Using a stochastic approach based on queueing theory, the authors in [19] analyze the packet distributions of interesting processes in the general case. In their work, they consider a P4 switch as data plane with feedback from the control plane, both represented by their own queues. They use different distributions and evaluate the packet distributions of important metrics such as the packet sojourn times. But, no explicit expressions are provided in [19] neither for the distribution nor for the mean of the sojourn time. Queueing networks with feedback have also been analyzed by other works in the literature, such as in [20], [21], [22]. However, common to [20], [21], and [22] is the fact that in their queueing networks with feedback the packet can go multiple times through the feedback branch (corresponding to the control plane in our case). This is different from our setup as we assume that the packet can go back to the controller at most once. Hence, these models are not suitable for our scenario.

Our work reconsiders the feedback-oriented queueing network model proposed in [9]. Furthermore, in our work the sojourn time of the system is re-evaluated accordingly based on different theoretical approximations for Erlang, exponential, and hyperexponential distributions, mimicking this way a wide spectrum of service time distributions in terms of variance. In addition, our work goes one step further and does not only consider one data plane device, but multiple.

Finally, this work is an extension of [1], where we provide the analysis for a single data plane and a controller. As a controller can be used for multiple switches, generalizing the model to multiple data planes, what we do in this work, is more realistic, hence, captures more reliably the behavior in such a network.

3. P4 Background

Network programmability was first introduced with the Software-Defined Network (SDN) architecture [23]. SDN separates the control plane from the data plane, introduces a centralized controller entity between the two planes, enables programmability at the control plane, and defines an interface (such as the OpenFlow protocol [24]) to push messages to the data plane, instructing it how to forward packets. This approach to programming networks is found to be limited since it is bound to the predefined protocols and actions supported by the OpenFlow architecture and protocol. To address these issues, P4 programmability was introduced to push programmability to the data plane, where the device's behavior can be customized independently from any predefined headers or actions (as in the OpenFlow case).

The latest P4 version, which is $P4_{16}$, separates the syntax of the language from the details of the hosting target, which is included in the target's architecture description. This P4 architecture describes the programmable blocks on the target and the interfaces to program them. This makes the P4 language target-independent so that it can be used to program any type of packet processor with a given architecture description.

The P4 programmability is mainly concerned with defining the packet processing behavior at the data plane of a device. Although the processing behavior is described based on the match-action abstraction (similarly to OpenFlow), unlike the OpenFlow case, P4 allows for defining arbitrary protocols and custom actions. When a packet arrives at a P4 data plane, it goes through a sequence of processing operations defined in the P4 program. First, the headers of the packet are extracted in the parser stage, which is described as a finite-state machine. Next, the ingress and egress processing stages in the P4 program transform the headers of the packet according to the defined tables in the program. Each table is defined based on a list of matching keys, and a list of possible custom actions that can be invoked upon matching. These actions are customizable and written as functions.

The P4 program defines the pipeline that a packet can go through, especially in terms of the tables that will be applied. It is the responsibility of the control plane to populate the tables in this pipeline with the rules that achieve the intended behavior of the use case application. If a packet arriving at a P4 data plane does not find a matching rule in the visited tables, it is forwarded to the control plane, which takes care of processing this packet. Then, it forwards this packet back to the data plane, and also sends a rule

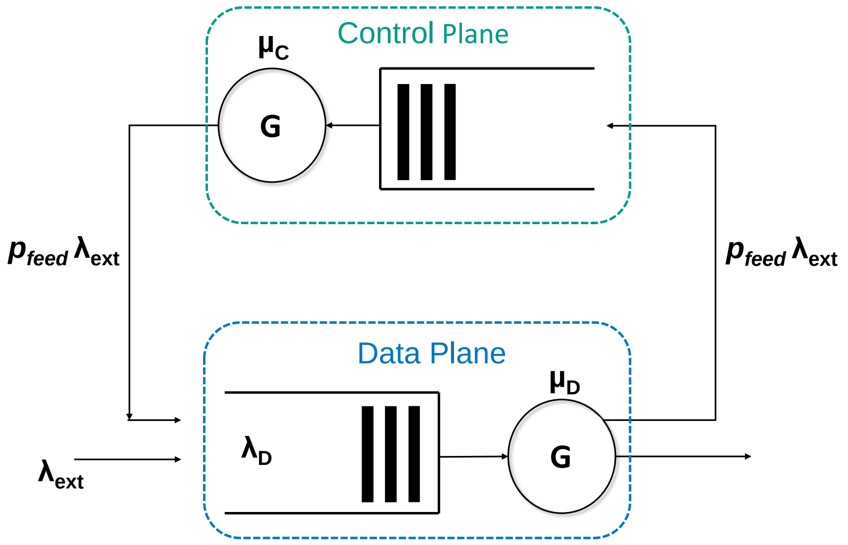


Figure 1: Queueing model for a P4 forwarding data plane with controller feedback.

to instruct following packets how to be processed. Accordingly, a packet can either get processed only at the data plane, or it could be processed at the data plane followed by the control plane followed by the data plane again. This life cycle of packets in P4-based systems guides the abstraction of the behavior of these systems as a *feedback-oriented system*, which is analyzed in the remainder of this paper.

4. P4 Performance Model: Single data plane case

In this section, we describe the performance model for P4 devices with controller feedback, followed by a detailed description of the data plane and control plane.

4.1. System Description

In SDN, the data plane is separated from the control plane. The former is responsible for the processing of packets, whereas the latter defines *how* they (i.e., packets) are processed. In case of a programmable data plane, the packet handling can be described with languages such as P4. The SDN paradigm allows one controller to control several data planes. Therefore, we consider this scenario as well later in Section 6. As we need the results from the single data plane for the scenario with multiple data planes, we develop first the models for the single data plane case.

Modeling a network consisting of data and control planes before deployment enables to understand the network behavior and properly dimension the participating devices. One of the most important performance metrics in related use cases is the *mean packet sojourn time*, i.e., the overall time a packet spends in the system. In a first step, in this work, a network consisting of only one data plane with controller feedback is considered. Fig. 1 shows a model of such a system. The data plane (lower part) and the control plane (upper part) are both represented by their own queues, each consisting of a waiting queue (buffer) and a server. We consider a data plane which is programmable with P4.

The path a packet traverses in the system is as follows. First, it arrives to the system following a Poisson arrival process with rate λ_{ext}, i.e., the inter-arrival time between any two packets arrivals is exponentially distributed. Then, the packet is first processed in the data plane with a service rate of μ_D. As the data plane is P4-enabled, this packet processing is defined by a P4 program. If an entry of the match-action table matches the packet information, it leaves the system. On the other hand, we assume that with a probability of p_{feed} there is no table entry and in that case the packet is sent back to the controller, which implies an arrival rate of $p_{feed}\lambda_{ext}$ there. The control plane processes the packets with rate μ_C, adds a table rule in the data plane for that packet and sends the packet back to the data plane. After being processed by the data plane (for the second time), the packet leaves the system as now there is a matching table entry. Thus, a packet can be sent to the controller only once.

4.2. Data Plane Processing

Upon arriving to the data plane, the packet is stored in the buffer. We assume that the buffer size of the data plane is infinite and the service discipline is First Come First Served (FCFS). There are two types of packets arriving to the data plane: *external packets* entering the system at that moment (with rate λ_{ext}) and *feedback packets* entering the data plane from the controller, i.e., for the second time (with rate $p_{feed}\lambda_{ext}$). Therefore, the overall arrival rate at the data plane is

$$\lambda_D = \lambda_{ext} + p_{feed}\lambda_{ext} = (1 + p_{feed})\lambda_{ext}. \tag{1}$$

Even though the external arrival process follows a Poisson process, this is not the case for the overall data plane arrival process. This stems from the fact that the controller feedback process depends on the external arrival process, violating the requirement for *independent increments*, characteristic to Poisson processes, making the system behavior more involved.

The service times in the data plane follow a general distribution with a mean of $\mathbb{E}[S_D]$ (the corresponding service rate is thus $\mu_D = \frac{1}{\mathbb{E}[S_D]}$) and a standard deviation of σ_D. The average total time a packet spends in the data plane (the sum of the corresponding service time and queueing delay) is denoted by $\mathbb{E}[T_D]$.

4.3. Control Plane Processing

The control plane also has an infinite buffer size and follows the FCFS order of service, like the data plane. The arrival process is a fraction of the departure process of the data plane and has a rate of

$$\lambda_C = p_{feed} \cdot \lambda_{ext}. \tag{2}$$

The service times in the control plane follow a general distribution with a mean of $\mathbb{E}[S_C]$ (the corresponding service rate is thus $\mu_C = \frac{1}{\mathbb{E}[S_C]}$) and a standard deviation of σ_C. The average total time a packet spends in the control plane (the sum of the corresponding service time and queueing delay) is denoted by $\mathbb{E}[T_C]$.

Both queues together form a network, known as *queueing network with feedback* [20].

4.4. Packet Sojourn Time

As already mentioned, the mean packet sojourn time is an important metric of interest. Therefore, in this paper we look in detail in achievable system performance in terms of sojourn time.

Packets in the system either directly leave the system after being processed in the data plane or experience controller involvement also. Hence, the mean packet sojourn time $\mathbb{E}[T]$ consists of the time a packet spends at the data plane $\mathbb{E}[T_D]$ and the control plane $\mathbb{E}[T_C]$. This is expressed as

$$\mathbb{E}[T] = \mathbb{E}[T_D] + p_{feed} \left(\mathbb{E}[T_D] + \mathbb{E}[T_C] \right), \tag{3}$$

or equivalently,

$$\mathbb{E}[T] = (1 + p_{feed})\mathbb{E}[T_D] + p_{feed}\mathbb{E}[T_c]. \tag{4}$$

Analyzing queueing networks and obtaining closed-form solutions is possible only for Jackson networks [25], i.e., when all the processes in the system are characterized by the memoryless property. Given that we assume non-exponential service times at the data plane and general distribution at the controller, the queueing network at hand is not a Jackson network. Therefore, we cannot exploit the results from there (i.e., from Jackson's networks) in our system. The aforementioned assumptions and requirements make the derivation of exact analytical solutions in closed-form infeasible. Consequently, in this paper, we focus on comparing the prediction accuracy that three approximation approaches offer (see Section 5).

In order to cover a wide range of possible distributions, we analyze the approximations for three different distribution types on the data plane and control plane service times based on their *coefficient of variation* c_V, which is defined as the ratio of the standard deviation of a random variable and its mean, and is a measure of how dispersed the samples from the mean of a random variable are.

The first considered distribution is the Erlang distribution, which represents a distribution with low variability, for which it holds $c_V < 1$. The second distribution is the exponential (memoryless) distribution, for which $c_V = 1$. Finally, the hyper-exponential distribution (for which $c_V > 1$) represents the scenario with a heavy-tailed service time.

In the next section, we derive the equations for all three approximation approaches. Then, in Section 7 we use simulations to compare the theoretical (approximation) results to the actual (simulation) results for the different approaches, for different distributions of service times. Table 1 contains the notation used in the remainder of this paper.

Parameter	Description
λ_{ext}	External arrival rate arriving at the data plane
μ_D	Service rate of the data plane
p_{feed}	Forwarding probability to the controller
μ_C	Service rate of the controller
$\mathbb{E}[S_D]$	Mean data plane service time
σ_D	Data plane service time standard deviation
$\mathbb{E}[T_D]$	Mean total time in the data plane
λ_C	Overall arrival rate at the controller
$\mathbb{E}[S_C]$	Mean control plane service time
σ_C	Control plane service time standard deviation
$\mathbb{E}[T_C]$	Mean total time in the controller
$\mathbb{E}[T]$	Mean packet sojourn time in the system
e_D	Frequency of visits by each packet to the data plane
e_C	Frequency of visits by each packet to the control plane
ρ_D	Data plane utilization
ρ_C	Control plane utilization
$c_{D,A}$	Coefficient of variation of the data plane arrival process
$c_{C,A}$	Coefficient of variation of the control plane arrival process
$c_{D,S}$	Coefficient of variation of the data plane service process
$c_{C,S}$	Coefficient of variation of the controller service process
$\mathbb{E}[N_D]$	Mean number of packets at the data plane
$\mathbb{E}[N_C]$	Mean number of packets at the control plane
n	Number of devices in the data plane
$\lambda_{ext,i}$	External arrival rate at the ith data plane device
$\mu_{D,i}$	Service rate of the ith data plane device
$p_{l,i}$	Probability to leave after service at the ith data plane
$p_{feed,i}$	Data plane i forwarding probability to the controller
λ_i	Overall arrival rate at the ith data plane device
$\mathbb{E}[S_{D,i}]$	Mean service time of the ith data plane device
$\sigma_{D,i}$	Service time standard deviation of the ith data plane

$\mathbb{E}[T_{D,i}]$	Total packet time in the ith data plane
e_i	Frequency of visits to the ith data plane
$p_{0,i}$	Packet probability to enter externally to the ith device
$p_{j,i}$	Packet probability to be sent from the jth to the ith device
ρ_i	Data plane utilization of the ith device
$c_{i,A}$	Coefficient of variation of the ith data plane arrival process
$c_{i,S}$	Coefficient of variation of the ith data plane service process
$\mathbb{E}[N_i]$	Mean number of packets at the ith data plane device

Table 1: Notation

5. Approximation Approaches

In this section, we first present two state-of-the-art approaches to approximate the mean packet sojourn time. These are the *Diffusion approximation* and the *Modified Diffusion approximation*. Then, we decouple the two queues (data plane and control plane), and use the corresponding equations for two independent M/G/1 queues.

5.1. Diffusion Approximation

The first approximation has been introduced in [26] and [27]. The authors use the diffusion approximation technique to approximate the number of packets $\mathbb{E}[N]$ in a queueing system with generally distributed service times. With this approach, the probability of finding k packets in such a system with arrival rate λ, service rate μ, utilization $\rho = \frac{\lambda}{\mu}$, and coefficients of variation c_A for the arrival and c_S for the service process, can be approximated as:

$$\hat{\pi}(k) = \begin{cases} 1-\rho, & k=0 \\ \rho(1-\hat{\rho})\hat{\rho}^k - 1, & k>0 \end{cases} \tag{5}$$

where

$$\hat{\rho} = \exp\left(\frac{-2(1-\rho)}{\rho c_A^2 + c_S^2}\right). \tag{6}$$

As a stable queue is required, i.e., a queue with utilization ratio $\rho < 1$, for the mean number of packets we have

$$\mathbb{E}[N] = \sum_{k=1}^{\infty} k\hat{\pi}(k) = \frac{\rho}{1-\hat{\rho}}. \tag{7}$$

These results are now applied to our model (following the steps like in [28]) to find an approximation for the mean sojourn time assuming known mean and standard deviations of the service times. First, we need to introduce the parameters e_D and e_C, which denote the frequency of visits to the data plane and controller by each packet, respectively. Hence, $e_D = 1 + p_{feed}$ and $e_C = p_{feed}$.

The utilization at the data plane is

$$\rho_D = \frac{e_D \lambda_{ext}}{\mu_D}, \tag{8}$$

whereas at the control plane

$$\rho_C = \frac{e_C \lambda_{ext}}{\mu_C}. \tag{9}$$

With the coefficients of variations of the data plane service process $c_{D,S}$ and $c_{C,S}$ for the controller, the coefficients of variations for the arrival processes at the data plane $c_{D,A}$ and the controller $c_{C,A}$ can be obtained from the following expression [26]:

$$c_{i,A}^2 = 1 + \sum_{j=0}^{N} (c_{j,S}^2 - 1) \cdot p_{j,i}^2 \cdot \frac{e_j}{e_i}, \tag{10}$$

where N is set to 2 with $i \in \{D, C\}$ and

$$j = \begin{cases} j = 0, & \text{representing external arrival stream} \\ j = 1, & \text{representing the data plane} \\ j = 2, & \text{representing the control plane.} \end{cases} \tag{11}$$

Using the conditions $p_{0,D} = p_{2,D} = 1$ (every external packet and every packet from the controller have to go to the data plane), $p_{0,C} = p_{1,D} = p_{2,C} = 0$ (external packets cannot go directly to the control plane, a packet from the data plane cannot re-enter immediately the data plane, and a packet from the control plane cannot re-enter the control plane again), and $p_{1,C} = \frac{p_{feed}\lambda_{ext}}{(1+p_{feed})\lambda_{ext}} = \frac{p_{feed}}{1+p_{feed}}$ (the probability that a packet in the data plane is a feedback packet) as well as exploiting the fact that the coefficient of variation of the external Poisson arrival process is 1 ($c_{0,S} = 1$) because of the fact the inter-arrival times of packets are exponentially distributed, $c^2_{D,A}$ can be calculated as

$$c^2_{D,A} = 1 + (c^2_{C,S} - 1) \cdot \frac{e_C}{e_D}. \tag{12}$$

Similarly, for the control plane we have

$$c^2_{C,A} = 1 + (c^2_{D,S} - 1) \cdot \frac{p^2_{feed}}{(1 + p_{feed})^2} \cdot \frac{e_D}{e_C}. \tag{13}$$

Further,

$$\hat{\rho}_D = \exp\left(\frac{-2(1-\rho_D)}{\rho_D c^2_{D,A} + c^2_{D,S}}\right), \tag{14}$$

and

$$\hat{\rho}_C = \exp\left(\frac{-2(1-\rho_C)}{\rho_C c^2_{C,A} + c^2_{C,S}}\right). \tag{15}$$

The mean number of packets at the data plane $\mathbb{E}[N_D]$ and at the control plane $\mathbb{E}[N_C]$ are given by:

$$\mathbb{E}[N_D] = \frac{\rho_D}{1 - \hat{\rho}_D}, \tag{16}$$

and

$$\mathbb{E}[N_C] = \frac{\rho_C}{1 - \hat{\rho}_C}. \tag{17}$$

Using Little's law [10], Eq.(1) and Eq.(2), the mean sojourn time for the data plane $\mathbb{E}[T_D]$ and for the control plane $\mathbb{E}[T_C]$ can be calculated as:

$$\mathbb{E}[T_D] = \frac{\mathbb{E}[N_D]}{(1 + p_{feed})\lambda_{ext}}, \tag{18}$$

and

$$\mathbb{E}[T_C] = \frac{\mathbb{E}[N_C]}{p_{feed}\lambda_{ext}}. \tag{19}$$

Substituting Eq.(18) and Eq.(19) into Eq.(4), we obtain the approximated mean sojourn time using this approach.

The coefficient of variation for exponentially distributed service times is 1. For the special case of Erlang distributed service times with a service rate of μ with k stages, the squared coefficient of variation reduces to $\frac{1}{k}$ [10]. The squared coefficient of variation for a hyperexponential distribution with two branches, i.e., consisting of two parallel exponential distributions each with probability p and $1-p$ and rates μ_1 and μ_2, with mean $\frac{p}{\mu_1}+\frac{1-p}{\mu_2}$, is [10]

$$c_{v,hyper}^2 = \frac{2\left(\frac{p}{\mu_1^2}+\frac{1-p}{\mu_2^2}\right)}{\left(\frac{p}{\mu_1}+\frac{1-p}{\mu_2}\right)^2} - 1. \tag{20}$$

From Eq.(20), it can be shown that $c_{v,hyper} > 1$, implying that hyper-exponential distributions are heavy tailed.

For each of these special cases, we can obtain the corresponding mean sojourn times, by substituting the corresponding coefficients of variation in Eqs.(12)-(15).

5.2. Modified Diffusion Approximation

The Diffusion approximation presented in Section 5.1 is further modified in [29] and [30]. In particular, by adjusting the calculation of the mean number of packets to

$$\mathbb{E}[N] = \rho\left(1+\frac{\rho c_A^2 + c_S^2}{2(1-\rho)}\right), \tag{21}$$

the authors claim to achieve more precise results for regions of higher utilization. The calculation approach follows the same scheme as in Section 5.1, but with the average number of packets in the data plane $\mathbb{E}[N_D]$ and the average number of packets in the control plane $\mathbb{E}[T_C]$ now obtained as:

$$\mathbb{E}[N_D] = \rho_D\left(1+\frac{\rho_D c_{D,A}^2 + c_{D,S}^2}{2(1-\rho_D)}\right), \tag{22}$$

and

$$\mathbb{E}[N_C] = \rho_C\left(1+\frac{\rho_C c_{C,A}^2 + c_{C,S}^2}{2(1-\rho_C)}\right). \tag{23}$$

Depending on the distribution of the service time in the data plane and in the control plane, the actual values for $\mathbb{E}[N_D]$ and $\mathbb{E}[N_C]$ can be obtained from Eq.(22) and Eq.(23), respectively.

Finally, substituting Eq.(22) and Eq.(23) into Eq.(4), we obtain the approximate mean sojourn time with this approach.

5.3. M/G/1 Approximation

This approximation is based on the assumption that both queues are considered as mutually independent with corresponding Poisson arrival processes. Looking at each queue separately, the mean sojourn time $\mathbb{E}[T]$ can be calculated with the well-known expression for the average sojourn time in an M/G/1 queue [28]:

$$\mathbb{E}[T] = \frac{\rho}{1-\rho} \cdot \frac{\mathbb{E}[S]}{2} \cdot (c_S^2 + 1) + \mathbb{E}[S], \tag{24}$$

where $\mathbb{E}[S]$ represents the mean service time and c_S is the coefficient of variation of the service time. Thus, $\mathbb{E}[T_D]$ and $\mathbb{E}[T_C]$ used for the calculation of the mean sojourn time of the P4 device system are

$$\mathbb{E}[T_D] = \frac{\rho_D}{1-\rho_D} \cdot \frac{\mathbb{E}[S_D]}{2} \cdot (c_{D,S}^2 + 1) + E[S_D], \tag{25}$$

and

$$\mathbb{E}[T_C] = \frac{\rho_C}{1-\rho_C} \cdot \frac{\mathbb{E}[S_C]}{2} \cdot (c_{C,S}^2 + 1) + \mathbb{E}[S_C]. \tag{26}$$

Note that the higher the variance of the service time, the higher the mean time a packet spends in both data plane and control plane.

Finally, replacing Eq.(25) and Eq.(26) into Eq.(4), we obtain the mean sojourn time approximation with this approach.

6. Analyses for Multiple Data Planes

In this section, we describe the performance model for an SDN network consisting of multiple data plane devices and a single controller serving them. Then, the operation of data plane and the control plane are described in detail for a packet traversing through the network. This is followed by the analysis of the packet mean sojourn time in such a system. Further, the theoretical analysis for the three different approximation approaches is provided.

6.1. System Description

In Section 4, the queueing model for a P4 device with controller feedback was presented. However, SDN networks may consist of multiple data planes with controller feedback, which is more realistic. Namely, packets will be usually traversing through multiple forwarding devices until reaching the

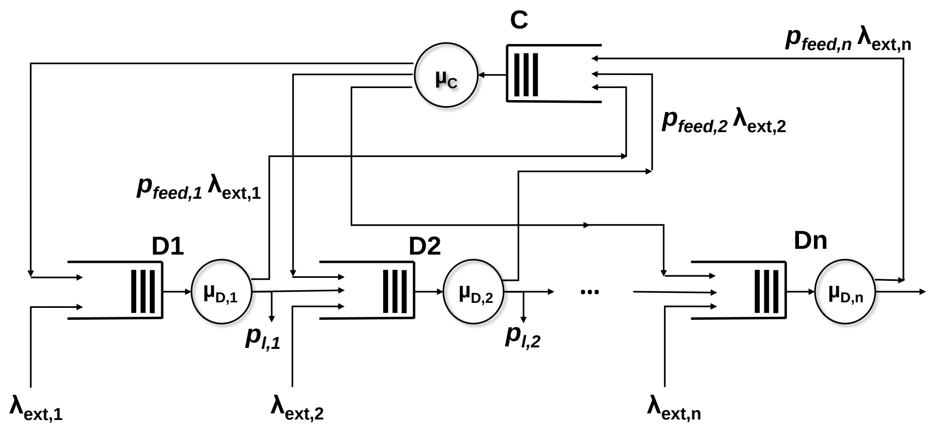


Figure 2: Queueing model for multiple P4 data plane devices with controller feedback.

destination. Fig. 2 illustrates such a system. Multiple data plane devices (lower part) are served by one controller (upper part). Again, each part of the data and control plane is modeled with a waiting queue and a server. The data plane devices are programmable with P4. There are in total n data planes.

In this model, the path of a packet is now as following: Packets arrive at the ith data plane device following a Poisson process with rate $\lambda_{ext,i}$. An arriving packet to that data plane is stored in the data plane queue. We assume that the queue sizes are infinite. Then, after waiting for some time until all the packets in front of it are processed, the aforementioned packet enters the data plane service. Again, we assume that the service discipline in all the devices is FCFS. The packet is served by that data plane with service rate $\mu_{D,i}$. After completion (the packet being completely processed), there are three different paths for that packet. This depends on the table entry for that packet. On the one hand, if a table rule exists, the packet is either forwarded to the next data plane device to continue its route, or leaves the system immediately. The latter happens in case the current data plane is directly linked to the destination. We assume this happens with probability $p_{l,i}$. In the nth queue, i.e., the last data plane in the sequence, the packet cannot be forwarded anymore but leaves the system immediately.

On the other hand, if a packet is not matched to a table entry, it is being forwarded to the control plane for further processing. This occurs with probability $p_{feed,i}$. This packet has to be queued if other packets are being processed at the controller. The service discipline at the controller is also FCFS. After the packet reaches the head of the queue and after that enters service, the controller processes the packet with a service rate of μ_C. As the controller has a top-level view of the whole network, it installs all required rules for the way the packet needs to follow in the corresponding switches. The packet is then sent back to the ith data plane device, where it is again processed. As there is a matching table rule now, the packet either leaves the system immediately or is forwarded to the next data plane device. The former happens with probability $p_{l,i}$ and the latter with probability $1 - p_{l,i}$. As the table entries of the following data plane devices have already been updated to match the packet, it is not forwarded again to the controller anymore. Thus, after being processed, it either leaves the system directly or is forwarded to the next data plane device. At the next data plane device, after being processed, it either leaves the system immediately with probability $p_{l,i+1}$ or goes to the next data plane (with probability $1-p_{l,i+1}$). The process continues until the packet leaves the system. This means that a packet can only go to the controller at most once and only from the data plane at which it arrived externally. After a packet goes to the controller, it is forwarded back to the data plane that sent it to the controller, going on until leaving the system.

6.2. Data Plane Processing

As already mentioned, each data plane device is modeled as an infinite queue, consisting of a waiting queue and service time. Packets arriving at the ith queue are either (i) *external packets* (with a rate of $\lambda_{ext,i}$), i.e., those that enter the system for the first time; (ii) *forwarded packets* from the previous data planes with a rate of $(1 - p_{l,i-1})\lambda_{ext,i-1} + (1 - p_{l,i-1})(1 - p_{l,i-2})\lambda_{ext,i-2} + \ldots + (1 - p_{l,i-1}) \ldots (1 - p_{l,1})\lambda_{ext,1}$; or (iii) *feedback packets* from the controller (with a rate of $p_{feed,i}\lambda_{ext,i}$). Thus, the total arriving rate at the ith data plane device is

$$\lambda_i = \lambda_{ext,i} + p_{feed,i}\lambda_{ext,i} + \sum_{j=1}^{i-1} \lambda_{ext,j} \prod_{k=j}^{i-1} (1 - p_{l,k}). \tag{27}$$

The service time of the ith data plane device follows a general distribution with mean $\mathbb{E}[S_{D,i}]$ and standard deviation $\sigma_{D,i}$. The corresponding service rate is thus $\mu_{D,i} = \frac{1}{\mathbb{E}[S_{D,i}]}$. The average time a packet spends in data plane i is denoted by $\mathbb{E}[T_{D,i}]$.

6.3. Control Plane Processing

Similar to the data plane, the control plane also has an infinite buffer and the packets are served in the FCFS order. The arrival rate at the control plane is equal to the sum of the arrival rates of feedback packets from each data plane:

$$\lambda_c = \sum_{i=1}^{n-1}(1 - p_{l,i})p_{feed,i}\lambda_{ext,i} + p_{feed,n}\lambda_n. \tag{28}$$

Two important things should be noted from the previous expression. First, the probability for the packet to leave the system immediately after the data plane i is $p_{l,i}$. Hence, the probability for a packet to be fed back to the controller is $(1 - p_{l,i})p_{feed,i}$, as that packet belongs to the group of packets that have not left immediately. Second, the packet after data plane n that is not sent back to the controller leaves the system (see Fig. 2). Hence, the probability term corresponding to data plane n in Eq.(28) is $p_{feed,n}$.

The service times at the controller undergo a general distribution with mean $\mathbb{E}[S_C]$, leading to a rate of $\mu_C = \frac{1}{\mathbb{E}[S_C]}$. The standard deviation of the service time at the controller is denoted by σ_C. The average total time a packet spends in the control plane (the sum of the corresponding service time and queueing delay) is denoted by $\mathbb{E}[T_C]$.

We assume that the data plane queues as well as the controller queue are all stable, i.e. $\rho < 1$. We are interested in analyzing the steady-state behavior of the system.

6.4. Packet Sojourn Time

For the prior introduced network model with n multiple data plane devices and a single controller, the mean sojourn time of a packet in the system is very important. In this paper, we consider the mean sojourn time of a packet arriving at the first data plane device (following a Poisson process with rate $\lambda_{ext,1}$) until it leaves the system after the ith queue (as already described, it can leave after being processed by switch 1, 2, ..., n). Thus, we need to determine the options of packet traversing the system until leaving and the corresponding probabilities.

Path 1. The packet leaves directly after being served in the first data plane device. The probability for this is $p_{l,1}$, and the experienced delay $\mathbb{E}[T_{D,1}]$.

Path 2. The packet is sent to the control plane after service completion in the first data plane device and then leaves the system directly after going through the data plane again. The probability for this is $(1-p_{l,1})p_{feed,1}p_{l,1}$ and the corresponding delay is $2\mathbb{E}[T_{D,1}]+\mathbb{E}[T_C]$.

Path 3. The packet is forwarded up to the ith data plane device sequentially without going through the controller, and after service completion in the i device leaves the system. The probability for this to be the case is $(1-p_{l,1})(1-p_{feed,1})(1-p_{l,2})\dots(1-p_{l,i-1})p_{l,i}$, and the expected delay for that packet is $\sum_{j=1}^{i}\mathbb{E}[T_{D,j}]$, $\forall i \in \{2,\dots,n-1\}$.

Path 4. The packet is forwarded through all data plane devices and leaves the system without ever visiting the controller. The probability in this case is $(1-p_{l,1})(1-p_{feed,1})(1-p_{l,2})\dots(1-p_{l,n-1})$, and the corresponding delay is $\sum_{i=1}^{n}\mathbb{E}[T_{D,i}]$.

Path 5. The packet is sent to the controller (from data plane 1) and then leaves the system after service completion in the ith queue. Note that the packet can go at most once to the controller, implying that it went through data planes 1 to $i-1$ sequentially, after going to the data plane 1 and the controller first. The probability for this outcome is $(1-p_{l,1})p_{feed,1}(1-p_{l,1})\dots(1-p_{l,i-1})p_{l,i}$. The expected delay in this case is $\sum_{j=1}^{i}\mathbb{E}[T_{D,j}]+\mathbb{E}[T_{D,1}]+\mathbb{E}[T_C]$, $\forall i \in \{2,\dots,n-1\}$.

Path 6. The packet goes to the controller from data plane 1 and then is forwarded all the way out, i.e., goes through all the data planes (1 to n) before leaving the system. So, it goes twice through data plane 1. The probability for this path is $(1-p_{l,1})p_{feed,1}(1-p_{l,1})\dots(1-p_{l,n-1})$, and the expected delay is $\sum_{i=1}^{n}\mathbb{E}[T_{D,i}]+\mathbb{E}[T_{D,1}]+\mathbb{E}[T_C]$.

Combining the probabilities and the average delays for each case in Path 1-Path 6, we obtain:

Result 1. *The average sojourn time of a packet in the system with n se-*

quential data planes and a single controller (shown in Fig. 2) is

$$
\begin{aligned}
\mathbb{E}[T] = p_{l,1}\mathbb{E}[T_{D,1}] + (1-p_{l,1})p_{feed,1}p_{l,1}\left(2\mathbb{E}[T_{D,1}] + \mathbb{E}[T_C]\right) + \\
\sum_{i=2}^{n-1}\left((1-p_{feed,1})p_{l,i}\prod_{j=1}^{i-1}(1-p_{l,j})\sum_{j=1}^{i}\mathbb{E}[T_{D,j}]\right) + \\
(1-p_{feed,1})\prod_{i=1}^{n-1}(1-p_{l,i})\sum_{i=1}^{n}\mathbb{E}[T_{D,i}] + \\
\sum_{i=2}^{n-1}\left((1-p_{l,1})p_{feed,1}p_{l,i}\prod_{j=1}^{i-1}(1-p_{l,j})\left(\sum_{j=1}^{i}\mathbb{E}[T_{D,j}] + \mathbb{E}[T_{D,1}] + \mathbb{E}[T_C]\right)\right) + \\
p_{feed,1}(1-p_{l,1})\prod_{i=1}^{n-1}(1-p_{l,i})\left(\sum_{i=1}^{n}\mathbb{E}[T_{D,i}] + \mathbb{E}[T_{D,1}] + \mathbb{E}[T_C]\right).
\end{aligned}
\tag{29}
$$

Having obtained the average sojourn time in the general form in Result 1, we need to determine average times the packet spends in each data plane $\mathbb{E}[T_{D,i}], \forall i \in 1, \dots, n$ and in the controller $\mathbb{E}[T_C]$. We compute these averages approximately using the three approaches which were also used in the case of a single data plane: Diffusion approximation, Modified Diffusion approximation, and Independent M/G/1 queues. We do this in the following three subsections. Then, the theoretical results of these approaches are compared with the actual (simulation) results and their prediction accuracy evaluated in Section 7.

6.5. Diffusion approximation

As we know from the single data plane case, when using the Diffusion approximation, first, we need to determine the frequency of visits e_i for all n forwarding devices, and e_C (nominally denoted here by the index $n+1$) for the controller. These are calculated as in [28] using

$$
e_i = p_{0,i} + \sum_{j=1}^{n+1} e_j p_{j,i}, \quad \forall i \in \{1, \dots, n+1\}, \tag{30}
$$

where $p_{0,i}$ is the probability that a packet entering the network from outside enters at the ith device and $p_{j,i}$ is the probability for a packet to be forwarded from the jth device to device i. To simplify the notation and analysis, we consider the case with two data planes.

For the considered network we need to determine the probabilities from Eq.(30). First, $p_{0,D1} = \frac{\lambda_{ext,1}}{\lambda_{ext,1}+\lambda_{ext,2}}$, and $p_{0,D2} = \frac{\lambda_{ext,2}}{\lambda_{ext,1}+\lambda_{ext,2}}$. Further, $p_{0,C} = 0$, as an external packet never goes directly to the controller. The probability for a packet after leaving data plane 1 to go to the controller is $p_{D1,C} = (1-p_{l,1})p_{feed,1}$, and for a packet from data plane 2, this probability is $p_{D2,C} = p_{feed,2}$. The probability for a packet leaving data plane 1 to go to data plane 2 is $p_{D1,D2} = (1-p_{l,1})(1-p_{feed,1})$. The probability for a transition from the controller to data plane 1 is

$$p_{C,D1} = \frac{(1-p_{l,1})p_{feed,1}\lambda_{ext,1}}{(1-p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}, \tag{31}$$

and from the controller to data plane 2 is

$$p_{C,D2} = \frac{p_{feed,2}\lambda_{ext,2}}{(1-p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}. \tag{32}$$

The numerator in Eq.(31) is the arrival rate of data plane 1 packets to the controller, whereas the numerator in Eq.(32) represents the arrival rate of packets arriving to the controller which externally came to data plane 2. The denominators in Eq.(31) and Eq.(32) represent the total arrival rate at the controller.

Substituting the previous probabilities into Eq.(30), for the frequency of visits to data plane 1 we obtain

$$e_{D1} = \frac{\lambda_{ext,1}}{\lambda_{ext,1}+\lambda_{ext,2}} + e_c \cdot \frac{(1-p_{l,1})p_{feed,1}\lambda_{ext,1}}{(1-p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}. \tag{33}$$

Similarly, for the frequency of packet visits to data plane 2, we have

$$e_{D2} = \frac{\lambda_{ext,2}}{\lambda_{ext,1}+\lambda_{ext,2}} + e_{D1}(1-p_{l,1})(1-p_{feed,1}) + e_c \cdot \frac{p_{feed,2}\lambda_{ext,2}}{(1-p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}, \tag{34}$$

whereas for the frequency of visits to the controller:

$$e_C = e_{D1}(1-p_{l,1})p_{feed,1} + e_{D2}p_{feed,2}. \tag{35}$$

Eqs.(33)-(35) comprise a system of linear equations with three unknowns e_{D1}, e_{D2}, and e_C. Hence, it has a unique solution in e_{D1}, e_{D2}, and e_C.

As a next step, we need to determine the arrival rates to the data planes and the controller. To that end, we adapt Eq.(27) to our case, and for the overall arrival rate at the first data plane, we have

$$\lambda_{D1} = (1 + (1 - p_{l,1})p_{feed,1})\lambda_{ext,1}, \tag{36}$$

whereas for the overall arrival rate to data plane 2, we obtain

$$\lambda_{D2} = (1 - p_{l,1})\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}. \tag{37}$$

Adapting Eq.(28) to our case, for the arrival rate at the controller, we get

$$\lambda_C = (1 - p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}. \tag{38}$$

In line with Eq.(8), the utilization of the (server of) data plane 1 is

$$\rho_{D1} = \frac{\lambda_{D1}}{\mu_{D1}} = \frac{(1 + (1 - p_{l,1})p_{feed,1})\lambda_{ext,1}}{\mu_{D1}}, \tag{39}$$

and for data plane 2 it is

$$\rho_{D2} = \frac{\lambda_{D2}}{\mu_{D2}} = \frac{(1 - p_{l,1})\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}{\mu_{D2}}. \tag{40}$$

As far as the utilization at the controller is concerned, in line with Eq.(8), we have

$$\rho_C = \frac{\lambda_C}{\mu_C} = \frac{(1 - p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}\lambda_{ext,2}}{\mu_C}. \tag{41}$$

With the coefficients of variation of the service times in each device denoted by $c^2_{D1,S}$, $c^2_{D2,S}$, and $c^2_{C,S}$, the squared coefficients of variation for the arrival processes for each device, using Eq.(10), are

$$c^2_{D1,A} = 1 + \sum_{j=0}^{N}(c^2_{j,S} - 1) \cdot p^2_{j,D1} \cdot \frac{e_j}{e_{D1}} = 1 + (c^2_{C,S} - 1)p^2_{C,D1}\frac{e_C}{e_{D1}}, \tag{42}$$

$$\begin{aligned} c^2_{D2,A} &= 1 + \sum_{j=0}^{N}(c^2_{j,S} - 1) \cdot p^2_{j,D2} \cdot \frac{e_j}{e_D 2} = \\ &= 1 + (c^2_{D1,S} - 1)p^2_{D1,D2} \cdot \frac{e_{D1}}{e_{D2}} + (c^2_{C,S} - 1)p^2_{C,D2} \cdot \frac{e_C}{e_{D2}}, \end{aligned} \tag{43}$$

and

$$\begin{aligned} c_{C,A}^2 &= 1 + \sum_{j=0}^{N} (c_{j,S}^2 - 1) \cdot p_{j,C}^2 \cdot \frac{e_j}{e_C} = \\ &= 1 + (c_{D1,S}^2 - 1) \cdot p_{D1,C}^2 \cdot \frac{e_{D1}}{e_C} + (c_{D2,S}^2 - 1) \cdot p_{D2,C}^2 \cdot \frac{e_{D2}}{e_C}, \end{aligned} \tag{44}$$

where for the input parameters, we substitute e_{D1}, e_{D2}, e_c, Eq.(31), Eq.(32), and other probabilities from the previous discussion into Eqs.(42)-(44).

Further, using Eq.(6) for the corresponding $\hat{\rho}$ for data plane 1, data plane 2, and the controller we have

$$\hat{\rho}_{D1} = \exp\left(\frac{-2(1-\rho_{D1})}{\rho_{D1} c_{D1,A}^2 + c_{D1,S}^2}\right), \tag{45}$$

$$\hat{\rho}_{D2} = \exp\left(\frac{-2(1-\rho_{D2})}{\rho_{D2} c_{D2,A}^2 + c_{D2,S}^2}\right), \tag{46}$$

and

$$\hat{\rho}_{C} = \exp\left(\frac{-2(1-\rho_{C})}{\rho_{C} c_{C,A}^2 + c_{C,S}^2}\right), \tag{47}$$

where ρ_{D1} is given by Eq.(39), ρ_{D2} by Eq.(40), ρ_C by Eq.(41), while $c_{D1,A}$, $c_{D2,A}$, and $c_{C,A}$ are obtained from Eqs.(42)-(44). Note that the coefficients of variation of the service times at each entity $c_{D1,S}$, $c_{D2,S}$, and $c_{C,S}$ are known because we know the corresponding distributions of the service times.

Having obtained ρ_{D1}, $\hat{\rho}_{D1}$, ρ_{D2}, $\hat{\rho}_{D2}$, ρ_C, and $\hat{\rho}_C$ previously, the average number of jobs at these devices, i.e., $\mathbb{E}[N_{D1}]$, $\mathbb{E}[N_{D2}]$, and $\mathbb{E}[N_C]$, can be obtained using the adjusted Eq.(7) as:

$$\mathbb{E}[N_{D1}] = \frac{\rho_{D1}}{1-\hat{\rho}_{D1}}, \tag{48}$$

$$\mathbb{E}[N_{D2}] = \frac{\rho_{D2}}{1-\hat{\rho}_{D2}}, \tag{49}$$

and

$$\mathbb{E}[N_{C}] = \frac{\rho_{C}}{1-\hat{\rho}_{C}}. \tag{50}$$

The mean sojourn time for a packet in each device can now be calculated using Little's law [10], Eq.(36), Eq.(37), and Eq.(38) as

$$\mathbb{E}[T_{D1}] = \frac{\mathbb{E}[N_{D1}]}{\lambda_{D1}} = \frac{\mathbb{E}[N_{D1}]}{(1 + (1 - p_{l,1})p_{feed,1})\lambda_{ext,1}}, \tag{51}$$

$$\mathbb{E}[T_{D2}] = \frac{\mathbb{E}[N_{D2}]}{\lambda_{D2}} = \frac{\mathbb{E}[N_{D2}]}{(1 - p_{l,1})\lambda_{ext,1} + (1 + p_{feed,2})\lambda_{ext,2}}, \tag{52}$$

and

$$\mathbb{E}[T_C] = \frac{\mathbb{E}[N_C]}{\lambda_C} = \frac{\mathbb{E}[N_C]}{(1 - p_{l,1})p_{feed,1}\lambda_{ext,1} + p_{feed,2}.\lambda_{ext,2}}. \tag{53}$$

For our case with two data planes and a single controller, Eq.(29) reduces to

$$\begin{aligned}\mathbb{E}[T] = {} & p_{l,1}\mathbb{E}[T_{D1}] + (1 - p_{l,1})(1 - p_{feed,1})(\mathbb{E}[T_{D1}] + \mathbb{E}[T_{D2}]) + \\ & + (1 - p_{l,1})p_{feed,1}p_{l,1}(2\mathbb{E}[T_{D1}] + \mathbb{E}[T_C]) + \\ & + (1 - p_{l,1})p_{feed,1}(1 - p_{l,1})(2\mathbb{E}[T_{D1}] + \mathbb{E}[T_C] + \mathbb{E}[T_{D2}]).\end{aligned} \tag{54}$$

Inserting Eq.(51), Eq.(52), and Eq.(53) into Eq.(54), the approximated packet average sojourn time using this approach can be obtained.

6.6. Modified Diffusion Approximation

The modified diffusion approximation approach is very similar to the diffusion approximation. The only difference is in the modified computation of the average number of packets in an entity $\mathbb{E}[N]$. In particular, for our considered network, in line with Eq.(21), they are obtained from

$$\mathbb{E}[N_{D1}] = \rho_{D1}\left(1 + \frac{\rho_{D1}c^2_{D1,A} + c^2_{D1,S}}{2(1 - \rho_{D1})}, \right), \tag{55}$$

$$\mathbb{E}[N_{D2}] = \rho_{D2}\left(1 + \frac{\rho_{D2}c^2_{D2,A} + c^2_{D2,S}}{2(1 - \rho_{D2})}\right), \tag{56}$$

and

$$\mathbb{E}[N_C] = \rho_C\left(1 + \frac{\rho_C c^2_{C,A} + c^2_{C,S}}{2(1 - \rho_C)}\right). \tag{57}$$

The mean sojourn time for a packet to pass each device using this approach can then be calculated by substituting Eq.(55) into Eq.(51) for data plane 1, Eq.(56) into Eq.(52) for data plane 2, and Eq.(57) into Eq.(53) for the controller.

Finally, those obtained values for $\mathbb{E}[T_{D1}]$, $\mathbb{E}[T_{D2}]$, and $\mathbb{E}[T_C]$ can be inserted into Eq.(54) to calculate the mean packet sojourn time in this network.

6.7. M/G/1 Approximation

With the third approximation approach that we use in this paper, the behavior of all the queues at all data planes and at the controller are assumed to be independent, with Poisson arrival process at each one of them with corresponding arrival rates. Note that this again is only an approximation as in reality these queues are dependent and the arrival processes are not Poisson. However, as will be shown in Section 7, this approach in most of the scenarios of interest provides the closest match to the actual (simulation) results for the multiple data plane network as well.

Using Eq.(24), for the average time a packet spends in data plane i, the approximation is given by

$$\mathbb{E}[T_{D,i}] = \frac{\rho_{D,i}}{1-\rho_{D,i}} \cdot \frac{\mathbb{E}[S_{D,i}]}{2} \cdot (c_{D,i,S}^2 + 1) + E[S_{D,i}], \tag{58}$$

where $\rho_{D,i} = \frac{\lambda_i}{\mu_{D,i}}$ is the utilization of data plane i, with λ_i obtained from Eq.(27). The parameter $\mathbb{E}[S_{D,i}]$ denotes the mean service time in data plane i, whereas $c_{D,i,S}$ is its coefficient of variation.

Similarly, for the average time a packet spends in the controller, we have the approximation

$$\mathbb{E}[T_C] = \frac{\rho_C}{1-\rho_C} \cdot \frac{\mathbb{E}[S_C]}{2} \cdot (c_{C,S}^2 + 1) + \mathbb{E}[S_C], \tag{59}$$

with $\rho_C = \frac{\lambda_C}{\mu_C}$ denoting the utilization ratio of the controller, where λ_C is obtained from Eq.(28). The parameter $\mathbb{E}[S_C]$ represents the mean of the service time in the controller, whereas $c_{C,S}$ is its coefficient of variation.

Finally, substituting Eq.(58), $\forall i \in \{1, \dots, n\}$, and Eq.(59) into Eq.(29), we obtain the mean packet sojourn time of a packet in this network with the M/G/1 approximation approach.

7. Evaluation

In this section, the different approximation techniques for the packet mean sojourn time are evaluated by comparison to simulation results. First, the simulation setup and varied parameters are described. Then, the results for the different combinations of data plane and control plane service time distributions are presented for the single and the multiple data plane devices. Finally, the analytical results of all approaches are compared in order to find a good model for the different operation regions.

7.1. Simulation Setup

In a first step, the setup for our evaluation is described. First, we consider the single data plane device. In a further step, a network consisting of two data plane devices and a controller is analyzed followed by a network consisting of three data plane devices with controller feedback.

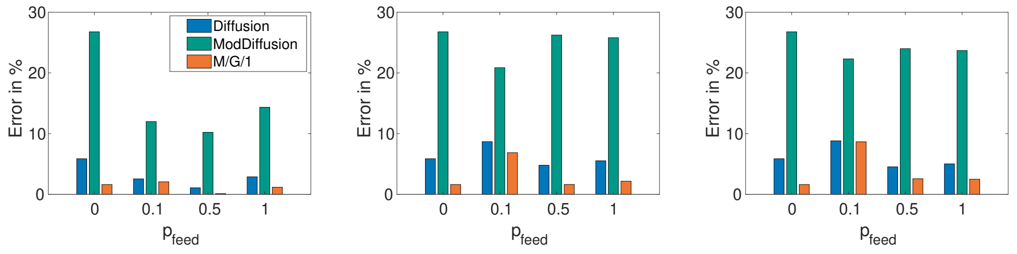
7.1.1. Single Data Plane Device

In order to evaluate the goodness of the analytical results for different regions of operations, the P4 device model with feedback is implemented in a packet-based MATLAB simulation and different parameters are varied. First, the data plane and control plane service times are either exponentially, Erlang or hyper-exponentially distributed, representing distributions with lower and higher variance. The mean service time of the data plane $\mathbb{E}[S_D] = 45.9\mu s$ is taken from [9] as the mean forwarding time for T4P4S running VxLAN. Additionally, the controller mean service times are also taken from [9]: $\mathbb{E}[S_C] = 31\mu s$ represents a controller which is faster, $\mathbb{E}[S_C] = 240\mu s$ which is medium, and $\mathbb{E}[S_C] = 10\,\text{ms}$ which is much slower compared to the data plane, i.e., it is comparable to an ONOS controller [31]. Whenever the Erlang distribution is used, the shape parameter (number of stages) is arbitrarily set to $k = 100$ in order to ensure a $c_{v,erl} = 0.1$. The hyperexponential distribution consists of two exponential distributions with rates μ_1 and μ_2. The probability to choose the first exponential distribution is $p = 0.9$, and $1 - p = 0.1$ otherwise. In order to ensure a $c_{v,hyper} > 1$, the following relation is set:

$$\mu_2 = 100\mu_1. \tag{60}$$

From Eq.(20) it can be shown that $c^2_{v,hyper} = 15.85$, independently of the used service time averages.

Another varied parameter is the probability for packets being sent to the controller, i.e., $p_{feed} \in \{0, 0.1, 0.5, 1\}$, showing the impact of the controller on the overall sojourn time. Each simulation is run for $100,000$ packets. For each of these cases, the controller's utilization is increased to a maximum of $\rho_c = 0.95$, always ensuring a stable controller queue. Also, a stable data plane is always ensured. The results obtained by those simulations are compared to the approximation results by taking the average error over the increasing load cases of the controller for one value of p_{feed}.



(a) Fast (b) Medium (c) Slow

Figure 3: Approximation errors for the three different controller speeds and the combination Exponential-Exponential.

7.1.2. Two Data Plane Devices

In this case, a network consisting of two data plane devices and a controller is simulated and the results are compared to the analytical results. Here, the same parameters as for the single data plane device network are applied. Additionally, we set $p_{feed,1} = p_{feed,2}$ and vary the probability of a packet to leave the system after the first data plane device, i.e., $p_{l,1} \in \{0, 0.1, 0.5\}$. The service times of both data plane devices are equal, i.e., $\mathbb{E}[S_{D1}] = \mathbb{E}[S_{D2}] = 45.9\mu s$.

7.1.3. Three Data Plane Devices

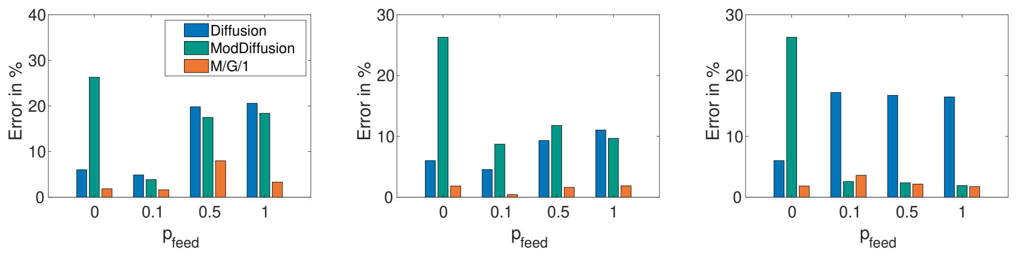
Now, a network with three data plane devices and a controller is considered and the results are compared to the theoretical approaches. The parameters are the same as for the single data plane device network. Additionally, the probabilities for a packet to visit the controller is the same, i.e., $p_{feed,1} = p_{feed,2} = p_{feed,3}$. Furthermore, the leaving probabilities of the data plane devices are the same and varied, i.e., $p_{l,1} = p_{l,2} \in \{0, 0.1, 0.5\}$. Finally, the mean service times of the data planes are again equal, i.e., $\mathbb{E}[S_{D1}] = \mathbb{E}[S_{D2}] = \mathbb{E}[S_{D3}] = 45.9\mu s$.

7.2. Single Data Plane Error Evaluation

In the following, the simulation results are compared to those obtained by the three approximation approaches for all combinations of service times distributions in the data plane and control plane.

7.2.1. Exponential-Exponential

Fig. 3 shows the errors of the different approximations for all controller processing speeds and a data plane as well as a control plane with expo-



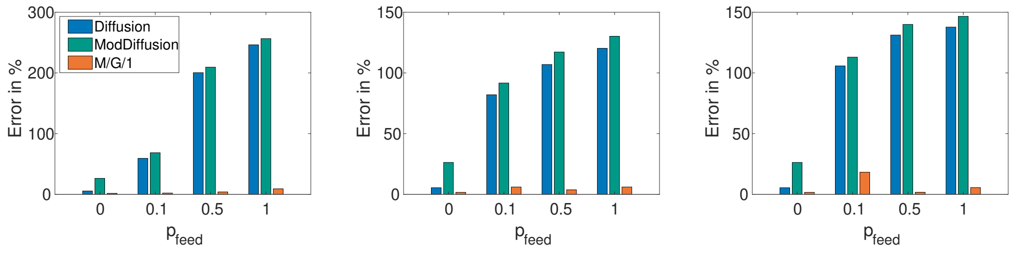
(a) Fast (b) Medium (c) Slow

Figure 4: Approximation errors for the three different controller speeds and the combination Exponential-Erlang.

nentially distributed service times, i.e., both distributions have $c_V = 1$. In the case of the fast controller (see Fig. 3a) and $p_{feed} = 0$, i.e., no controller interaction, the data plane can be modeled as an *M/G/1* queue and therefore the error of the closed-form equations for the *M/G/1* approach is very small. Additionally, the error results are independent of the controller speed. This also holds for other data plane distributions besides the exponential and when $p_{feed} = 0$. Hence, this case is not explained in the rest of this paper again. While the Modified Diffusion approach has high errors, the Diffusion approach is closer to the simulation and the *M/G/1* approximation is the best for all the values of p_{feed}. This behavior can also be observed for the medium (see Fig. 3b) and the fast controller (see Fig. 3c). In both cases, the Modified Diffusion approach approximates the simulation (actual) results significantly worse than the other two approaches. Note that for the *M/G/1* and the Diffusion approximation the highest error occurs for a low value of p_{feed}.

7.2.2. Exponential-Erlang

In the second case, the service times of the data plane are exponential and those of the control plane are Erlang-distributed. The approximation errors are shown in Fig. 4. For the fast controller case (see Fig. 4a), the Modified Diffusion approach has a high error compared to the other two. However, for higher values of p_{feed} the error is slightly smaller than with the Diffusion approach. The *M/G/1* approximation leads in all cases to the lowest errors. For the medium controller (see Fig. 4b), the behavior of the first two approximations change. While for low and medium values of p_{feed} the Diffusion approximation is better than the Modified Diffusion one, this



(a) Fast (b) Medium (c) Slow

Figure 5: Approximation errors for the three different controller speeds and the combination Exponential-Hyperexponential.

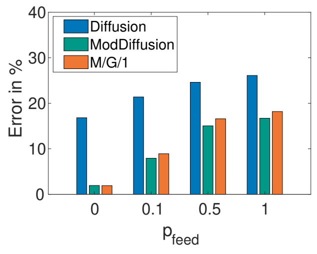
changes for $p_{feed} = 1$. In the slow controller case (see Fig. 4c), the Diffusion approach has a much higher error compared to the other two approximations. Note that for the case of $p_{feed} = 0.1$ the Modified Diffusion is better than the *M/G/1* approach.

7.2.3. Exponential-Hyperexponential

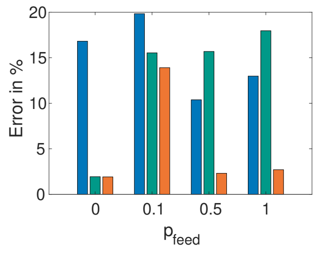
In this case, the data plane service times are exponential and those of the controller are hyperexponentially distributed, i.e., the control plane shows a high variance in its service times compared to the data plane. This impacts the errors of the Diffusion and the Modified Diffusion approach significantly (see Fig. 5). The errors for the fast (see Fig. 5a), the medium (see Fig. 5b) and the slow controller (see Fig. 5c) behave similarly. Even though the Modified Diffusion approach has the highest error for all values of p_{feed}, the Diffusion approximation is only slightly better. On the other side, the *M/G/1* approach produces very small errors in comparison to the other two approaches.

7.2.4. Erlang-Exponential

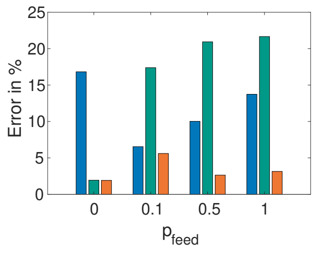
Fig. 6 shows the approximation errors for a data plane with Erlang and a controller with exponentially distributed service times. In the fast controller case (see Fig. 6a), the Diffusion approach error increases for higher values of p_{feed} and is the worst approximation. As opposed to most of the other cases, the *M/G/1* approach performs slightly worse than the Modified Diffusion. All errors increase with higher values of p_{feed}. The *M/G/1* approach is the best approximation again for all values of p_{feed} of the medium controller (see Fig. 6b). While the Diffusion approach performs worst for lower values of p_{feed}, the errors for higher values of p_{feed} are smaller than those of the



(a) Fast

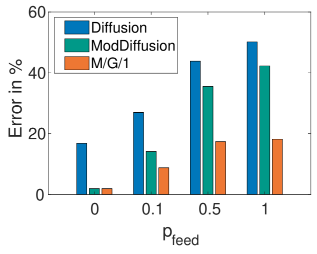


(b) Medium

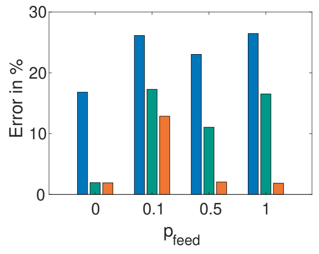


(c) Slow

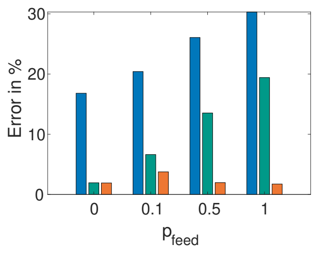
Figure 6: Approximation errors for the three different controller speeds and the combination Erlang-Exponential.



(a) Fast



(b) Medium



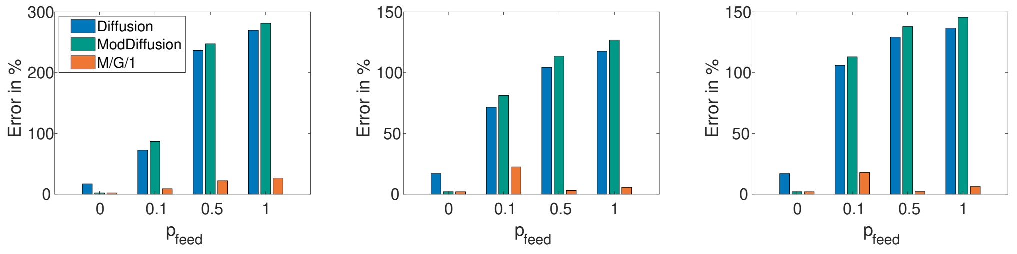
(c) Slow

Figure 7: Approximation errors for the three different controller speeds and the combination Erlang-Erlang.

Modified Diffusion approach. For the slow controller (see Fig. 6c) and $p_{feed} > 0$, the error for the Diffusion approach increases for higher values of p_{feed}, but is still smaller than the error from the Modified Diffusion approach. The *M/G/1* approach performs the best, with decreasing errors for increasing values of p_{feed}.

7.2.5. Erlang-Erlang

Fig. 7 presents the approximation errors for the case where the service times of both data plane and control plane follow the Erlang distribution. For the fast controller case (see Fig. 7a), all errors increase with higher values of p_{feed}. The best performing approximation is the *M/G/1* approach; the worst is the Diffusion approximation. The same holds for the medium controller (see Fig. 7b). However, as the error of the Diffusion approach is only slightly changing for higher values of p_{feed}, that of the *M/G/1* approximation decreases. The slow controller (see Fig. 7c) shows the same behavior as the fast controller with the exception of decreasing errors for increasing



(a) Fast (b) Medium (c) Slow

Figure 8: Approximation errors for the three different controller speeds and the combination Erlang-Hyperexponential.

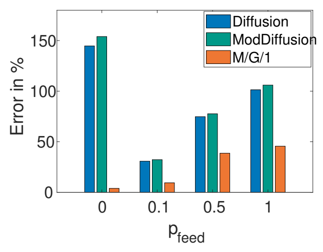
values of p_{feed} with the *M/G/1* approximation.

7.2.6. Erlang-Hyperexponential

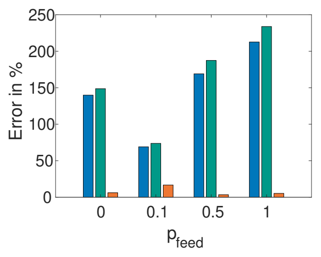
In this case, the service times of the data plane are Erlang and those of the controller are hyperexponentially distributed. The high variance of the control plane service times have a high impact on the errors of the Diffusion and the Modified Diffusion approach (see Fig. 8). The cases of the fast (see Fig. 8a), the medium (see Fig.8b), and the slow controller (see Fig. 8c) show the same behavior for the approximation errors. All errors are increasing for higher values of p_{feed} with all approaches except for *M/G/1* and a medium and slow controller, where the error first decreases for medium values of p_{feed} and then increases again. The Modified Diffusion approach performs slightly worse than the Diffusion approach, but significantly worse than the *M/G/1* approximation.

7.2.7. Hyperexponential-Exponential

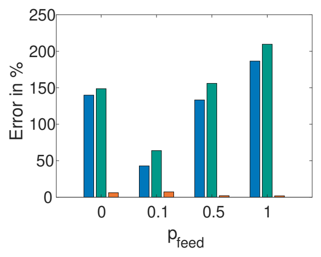
Fig. 9 shows the approximation errors for a data plane with hyperexponential and a control plane with exponentially distributed service times. The high impact of the variance of the data plane service times can be observed for the fast controller (see Fig. 9). Without controller involvement, i.e., $p_{feed} = 0$, the errors for the Diffusion and the Modified Diffusion approach are very high, whereas the *M/G/1* approximation is very close to simulation results. For higher values of p_{feed}, the errors also increase with the Diffusion approach, performing slightly better than the Modified Diffusion approximation. For all values of p_{feed}, the *M/G/1* approximation performs the best. Note that the controller involvement decreases the error for the Diffusion and Modified Diffusion approach. A similar behavior can be observed for the



(a) Fast

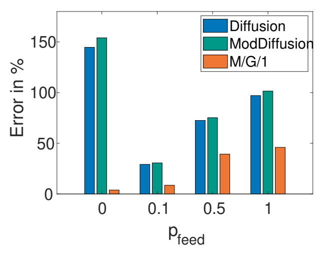


(b) Medium

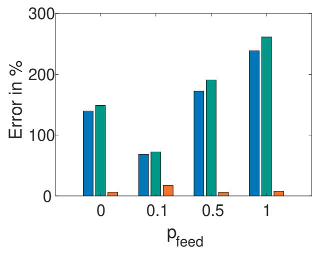


(c) Slow

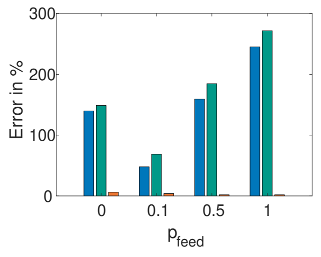
Figure 9: Approximation errors for the three different controller speeds and the combination Hyperexponential-Exponential.



(a) Fast



(b) Medium



(c) Slow

Figure 10: Approximation errors for the three different controller speeds and the combination Hyperexponential-Erlang.

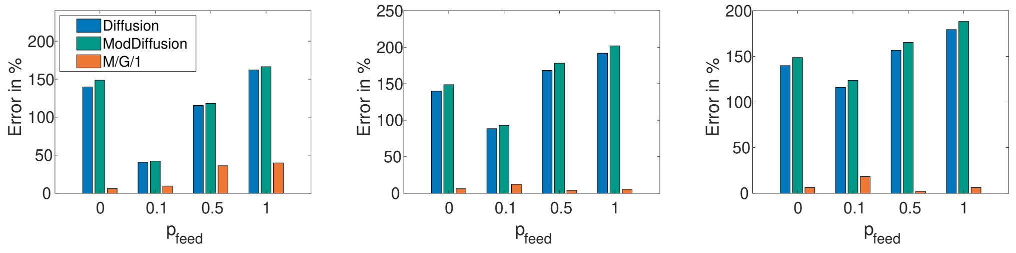
medium (see Fig. 9b) and slow controller (see Fig. 9c) with the exception of decreasing errors for the *M/G/1* approximation for higher values of p_{feed}, almost tending to 0. Additionally, the errors for higher values of p_{feed} for the Diffusion and Modified Diffusion approach grow larger than for $p_{feed} = 0$.

7.2.8. Hyperexponential-Erlang

In this case, the service times of the data plane are hyperexponential and those of the control plane are Erlang distributed. The approximation errors are shown in Fig. 10. The behavior of the errors for the fast (see Fig. 10a), the medium (see Fig. 10b), and the slow controller (see Fig. 10c) is similar to the case of a data plane with hyperexponential and a control plane with exponentially distributed service times, and is described there.

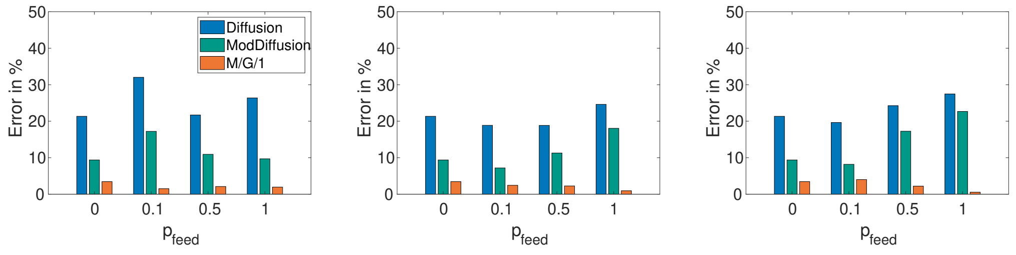
7.2.9. Hyperexponential-Hyperexponential

Fig. 11 shows the approximation errors for the last case, where the service times of both the data plane and the controller are hyperexponentially



(a) Fast (b) Medium (c) Slow

Figure 11: Approximation errors for the three different controller speeds and the combination Hyperexponential-Hyperexponential.



(a) Fast (b) Medium (c) Slow

Figure 12: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, Erlang distributed service times in the controller, and $p_{l,1} = 0$.

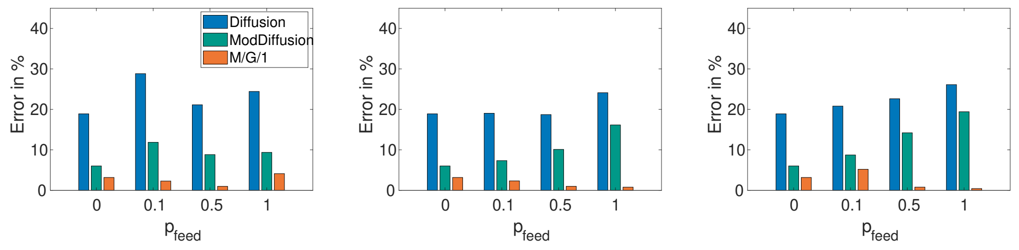
distributed. Again, the behavior for the fast (see Fig. 11a), the medium (see Fig. 11b), and the slow controller (see Fig. 11c) is similar to the Hyperexponential-Exponential case, and is described there.

7.3. Error Evaluation of Two Data Plane Devices

In the next scenario, one more data plane device is added to the network. Thus, the network now consists of two data plane devices and one controller. In the following, the errors of the three approximation approaches compared to simulation results are shown for four selected cases.

7.3.1. Erlang-Erlang-Erlang

The first case, where the service times of all devices in the network are Erlang distributed and $p_{l,1} = 0$, is shown in Fig. 12 for all three controller speeds. For the fast controller case in Fig. 12a, the *M/G/1* has the smallest error for all values of $p_{feed,1}$. Even though the error for the Modified



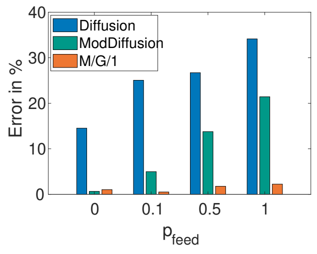
(a) Fast (b) Medium (c) Slow

Figure 13: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, Erlang distributed service times in the controller, and $p_{l,1} = 0.1$.

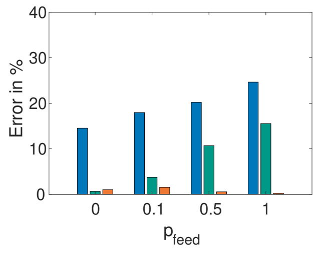
Diffusion is higher than for the *M/G/1*, it is still better than the Diffusion approximation in every case. This behavior can also be observed in Fig. 12b and Fig. 12c. Note that the error values for $p_{feed,1} = 0$ are the same across all three controller speeds, as for $p_{feed,1} = 0$ no packet is forwarded to the controller and the network simplifies to two sequential data plane devices. This holds also for the rest of the cases and is thus not mentioned in the remainder of the paper anymore.

Increasing the value of $p_{l,1}$ to 0.1, as in Fig. 13, decreases the error with the Diffusion and the Modified Diffusion approach for the fast controller case (see Fig. 13a), but increases it for the *M/G/1* approach, except for $p_{feed,1} = 0.5$. However, the *M/G/1* approximation still outperforms the other two approaches. For the medium and slow controller, shown in Fig. 13b and Fig. 13c, respectively, the error values with all three approaches are only slightly different compared to the case with $p_{l,1} = 0$.

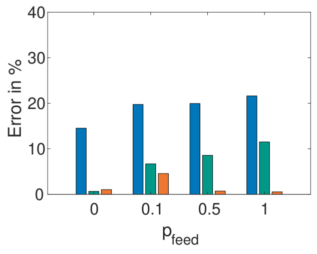
However, when further increasing $p_{l,1}$ (as shown in Fig. 14), this behavior changes. Considering a fast controller (Fig. 14a) with $p_{feed,1} = 0$, the Modified Diffusion approximation is slightly better than the *M/G/1* approach. With increasing values of $p_{feed,1}$, its error increases as well, whereas the error of the *M/G/1* approach only changes slightly. The Diffusion approach also shows increasing errors for increased values of $p_{feed,1}$, but it is significantly worse than the other two approaches. For the medium (Fig. 14b and the fast controller (Fig. 14b), the behavior is similar, except for the higher error peak for the *M/G/1* approach when $p_{feed,1} = 0.1$.



(a) Fast

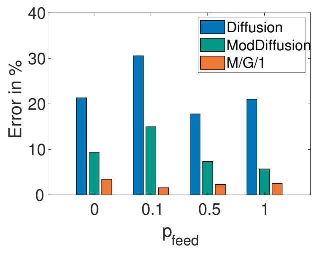


(b) Medium

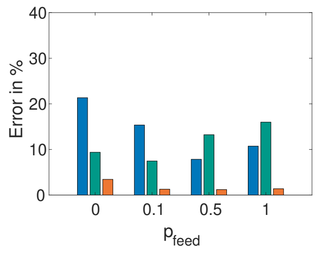


(c) Slow

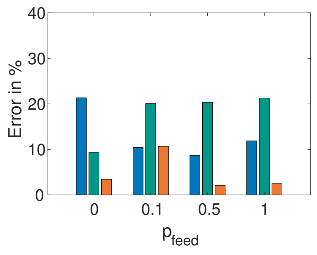
Figure 14: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, Erlang distributed service times in the controller, and $p_{l,1} = 0.5$.



(a) Fast



(b) Medium

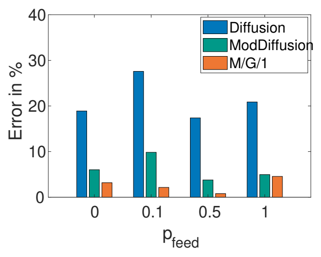


(c) Slow

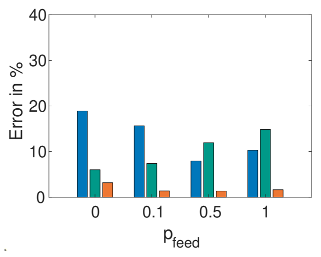
Figure 15: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, exponentially distributed service times in the controller, and $p_{l,1} = 0$.

7.3.2. Erlang-Erlang-Exponential

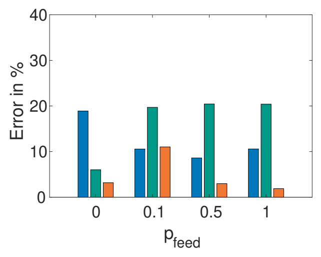
Fig. 15 shows the error results for the second case with $p_{l,1} = 0$. The data plane service times are both Erlang distributed, whereas the service time distribution of the controller follows an exponential distribution. For the fast controller case (Fig. 15a), the *M/G/1* approximation only varies slightly from the results obtained via simulation. The Diffusion and the Modified Diffusion are further away from the simulation result, whereas the Modified Diffusion Approach always outperforms the Diffusion approach. This behavior changes for the medium controller in Fig. 15b. While the approximation obtained with the *M/G/1* approach is still very close to the simulation results, the Diffusion approach shows better results than the Modified Diffusion for higher values of $p_{feed,1}$. For the slow controller case (Fig. 15c), a similar behavior for higher values of $p_{feed,1}$ can be observed. However, for $p_{feed,1} = 0.1$ the Diffusion approximation is slightly better than the *M/G/1* approach, which



(a) Fast

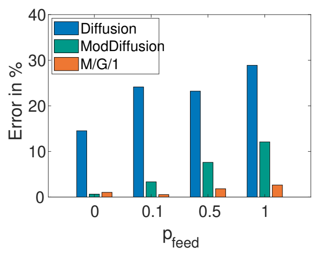


(b) Medium

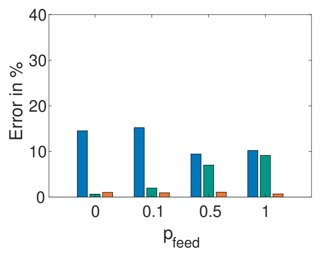


(c) Slow

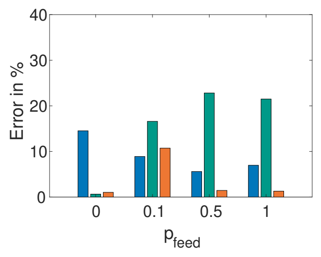
Figure 16: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, exponentially distributed service times in the controller, and $p_{l,1} = 0.1$.



(a) Fast



(b) Medium



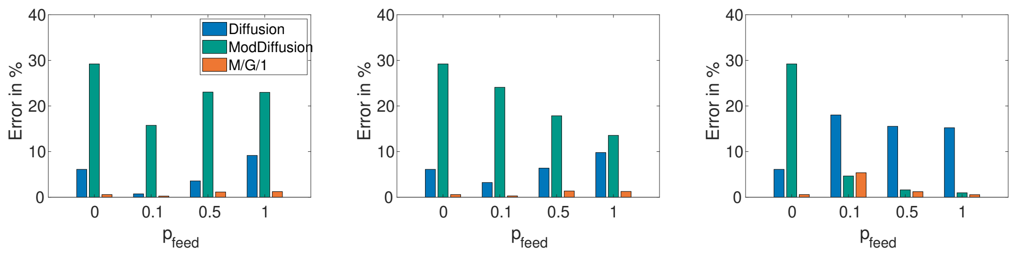
(c) Slow

Figure 17: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, exponentially distributed service times in the controller, and $p_{l,1} = 0.5$.

shows a high peak compared to the other controller speeds, and higher values of $p_{feed,1}$. This highest error value of the *M/G/1* approach for $p_{l,1} = 0$ is higher compared to the highest error in the previous case, i.e., when all service times are Erlang distributed. This is due to the higher c_v of the exponential distribution, which leads to increased variation in the service times results. Except for the case when no controller is involved, the Modified Diffusion approach performs the worst in the fast controller case.

When the value of $p_{l,1} = 0$ is increased to 0.1 (Fig. 16), the behavior and the errors of the three approximations only slightly change compared to the previous case with $p_{l,1} = 0$ for the fast (Fig. 16a), medium (Fig. 16b), and slow controller (Fig. 16c).

This changes for the results of $p_{l,1} = 0.5$ (see Fig. 17). In the fast controller case, shown in Fig. 17a, the Modified Diffusion approximation shows slightly better results than the *M/G/1*. However, its errors are increasing



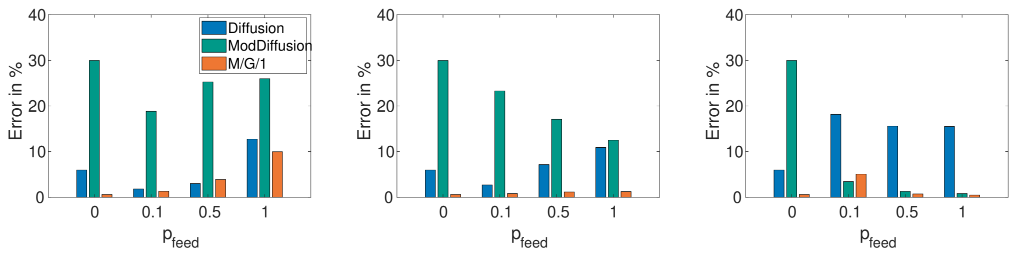
(a) Fast (b) Medium (c) Slow

Figure 18: Approximation errors for the three different controller speeds, exponentially distributed service times in the data plane, Erlang distributed service times in the controller, and $p_{l,1} = 0$.

faster for higher values of $p_{feed,1}$, whereas the errors of the *M/G/1* approach only slightly change and therefore, it is worse. The Diffusion approximation performs the worst of all approaches. Looking at the medium controller (Fig. 17b), the *M/G/1* approximation is very close to the simulation. The Modified Diffusion approach is very close to the *M/G/1* approximation for $p_{feed,1} = 0.1$, but the error increases for increased values of $p_{feed,1}$. While the Diffusion approach is significantly worse than the two others for $p_{feed,1} = 0.1$, the error decreases for increased values of $p_{feed,1}$ and is only slightly higher than the results of the Modified Diffusion approach. When the controller is slow (Fig. 17c), the behavior of the errors of the three approaches is very similar to the slow controller case for $p_{l,1} = 0$ (Fig. 15c), and is described there.

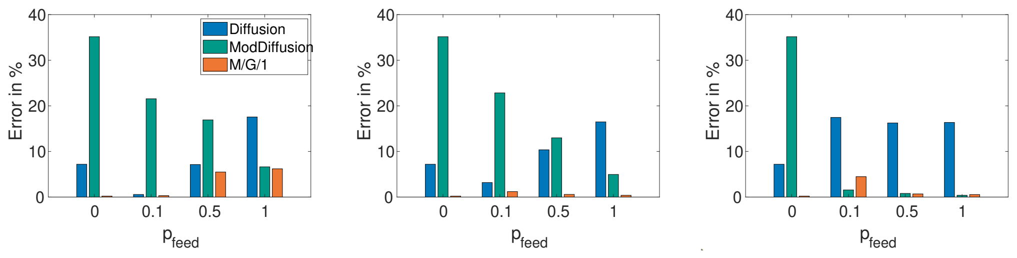
7.3.3. Exponential-Exponential-Erlang

In the third case, the service times in the data plane devices are exponential and in the controller are Erlang distributed. The results for $p_{l,1} = 0$ are shown in Fig. 18. The results for the fast (Fig. 18a) and the medium controller (Fig. 18b) are very similar. While the *M/G/1* approximation is very close to the simulation for all values of $p_{feed,1}$, the Diffusion approximation is also very close for medium values of $p_{feed,1}$. For $p_{feed,1} = 0$ and higher values of $p_{feed,1}$, the error increases a bit. The Modified Diffusion approach shows the worst results in all cases, even though the error approaches the error of the Diffusion approach for the medium controller and higher values of $p_{feed,1}$. In the slow controller case (Fig. 18c), the error of the Modified Diffusion and the *M/G/1* approximation is very small. Only for $p_{feed,1} = 0.1$ is the error of the former approach slightly smaller than the error of the latter. If



(a) Fast (b) Medium (c) Slow

Figure 19: Approximation errors for the three different controller speeds, exponentially distributed service times in the data plane, Erlang distributed service times in the controller, and $p_{l,1} = 0.1$.



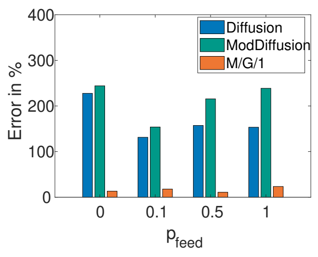
(a) Fast (b) Medium (c) Slow

Figure 20: Approximation errors for the three different controller speeds, Erlang distributed service times in the data plane, exponentially distributed service times in the controller, and $p_{l,1} = 0.5$.

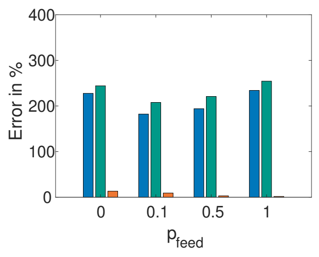
the controller is involved in the network, the Diffusion approximation shows significantly higher errors than the other two approaches.

For $p_{l,1} = 0.1$ (Fig. 19), the results for the fast (Fig. 19a), the medium (Fig. 19b), and the slow controller (Fig. 19) are very similar to the corresponding cases with $p_{l,1} = 0$. However, for the *M/G/1* approach, with the fast controller with $p_{feed,1} = 1$ the error is significantly increased compared to the slower controller with $p_{l,1} = 0$.

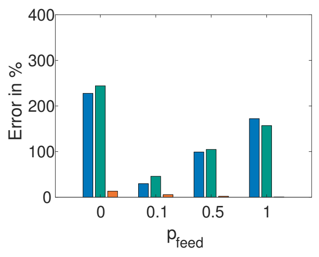
With $p_{l,1} = 0.5$ (as shown in Fig. 20), the results for the medium (Fig. 20b) and slow controller (Fig. 20c) are behaving similarly to the corresponding cases with $p_{l,1} = 0$ and $p_{l,1} = 0.1$. For the fast controller (Fig. 20a), the *M/G/1* approximation still shows the smallest error. It increases for higher values of $p_{feed,1}$ though. The Diffusion approach performs similarly to the *M/G/1* for small and medium values of $p_{feed,1}$; for higher values, the error increases. While the error for the Modified Diffusion Approach is very high



(a) Fast



(b) Medium



(c) Slow

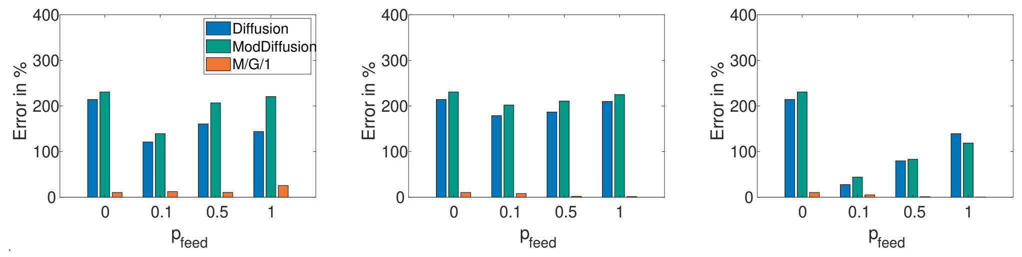
Figure 21: Approximation errors for the three different controller speeds, hyperexponentially distributed service times in the first and Erlang distributed in the second data plane device, Erlang distributed service times in the controller and $p_{l,1} = 0$.

compared to the error of the other two approaches, it performs almost as good as the *M/G/1* and better than the Diffusion approach for $p_{feed,1} = 1$.

7.3.4. Hyperexponential-Erlang-Erlang

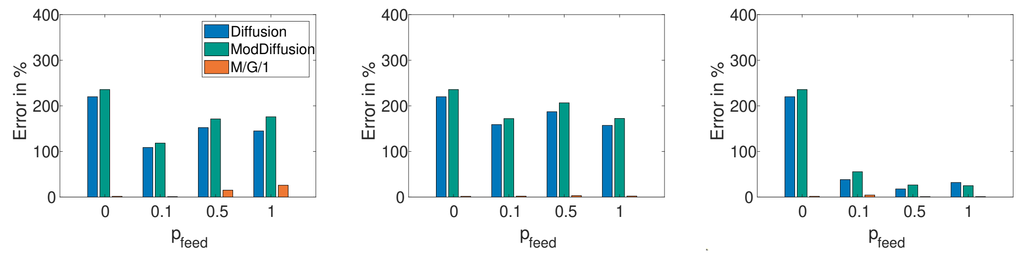
In the last case, the service times of the first data plane device are hyperexponentially distributed and those of the second device follow an Erlang distribution. The service times of the controller are also Erlang distributed. The results for $p_{l,1} = 0$ are shown in Fig. 21. For all three controller speeds, the fast (Fig. 21a), the medium (Fig. 21b), and the slow (Fig. 21c) one, the Diffusion and the Modified Diffusion approach are significantly worse than the *M/G/1* approach. For the fast controller case, the errors of the *M/G/1* approach are in general larger than for the medium and the slow controller. This is due to the impact of the slower controller to the latter cases. If the controller is fast compared to the data plane, the variance of the hyperexponentially distributed service times greatly influences the overall mean sojourn time. However, the slower the controller is, the higher the impact of the controller processing time, which diminishes the influence of the variance of the hyperexponential distribution on the overall mean sojourn time. The same effect can be observed when increasing the values of $p_{feed,1}$, starting with $p_{feed,1} = 0$ for the medium and the slow controller. The more the slower controller involved is, the lower the impact of the hyperexponentially distributed service times on the first data plane device.

The results of the considered case with $p_{l,1} = 0.1$ are shown in Fig. 22. The errors and their behavior of the three approximation approaches for the fast (Fig. 22a), the medium (Fig. 22b), and the slow controller (Fig. 22c) are very similar to the case with $p_{l,1} = 0$.



(a) Fast (b) Medium (c) Slow

Figure 22: Approximation errors for the three different controller speeds, hyperexponentially distributed service times in the first and Erlang distributed in the second data plane device, Erlang distributed service times in the controller, and $p_{l,1} = 0.1$.



(a) Fast (b) Medium (c) Slow

Figure 23: Approximation errors for the three different controller speeds, hyperexponentially distributed service times in the first and Erlang distributed in the second data plane device, Erlang distributed service times in the controller, and $p_{l,1} = 0.5$.

Finally, increasing $p_{l,1}$ to 0.5 results in very low errors for the *M/G/1* approach in almost all cases (Fig. 23). Only for the fast controller (Fig. 23a), the error goes up to 26% for higher values of $p_{l,1}$. The results of the Diffusion and the Modified Diffusion approach still differ from the simulation significantly in all cases. However, the error is greatly reduced in all cases when the slow controller is involved.

7.4. Error Evaluation for Three Data Plane Devices

Now, one more data plane device is added. The network now consists of three data plane devices and a controller. In the following, the error results using the *M/G/1* approximation for two selected cases are presented. We do this since the M/G/1 approach showed the best performance in the vast majority of the considered scenarios in this paper.

7.4.1. Erlang Distributed Data and Control Plane Service Times

		$p_{feed,i} = 0$	$p_{feed,i} = 0.1$	$p_{feed,i} = 0.5$	$p_{feed,i} = 1$
Slow Controller	$p_{l,i} = 0$	6.1%	3.3%	1.8%	1.0%
	$p_{l,i} = 0.1$	5.4%	1.3%	0.6%	0.6%
	$p_{l,i} = 0.5$	1.2%	2.9%	1.1%	0.9%
Medium Controller	$p_{l,i} = 0$	6.1%	5.4%	2.1%	1.4%
	$p_{l,i} = 0.1$	5.4%	5.3%	1.4%	0.8%
	$p_{l,i} = 0.5$	1.2%	3.2%	0.4%	0.4%
Fast Controller	$p_{l,i} = 0$	6.1%	7.5%	2.2%	1.6%
	$p_{l,i} = 0.1$	5.4%	7.5%	5.2%	3.4%
	$p_{l,i} = 0.5$	1.2%	5.6%	3.6%	4.7%

Table 2: Analytical errors compared to simulation results for three data plane devices and one controller, each with Erlang distributed service times.

In the first case, the service times of all devices, data plane and control plane, are Erlang distributed. Table 2 shows the approximation errors of the *M/G/1* approach for all controller speeds in all considered cases. The error is very low in the slow controller case for all values of $p_{feed,i}$ and $p_{l,i}$. Compared to that, the errors increase slightly for the medium controller and further for the fast controller. The errors are highest for all three controller speeds when the controller is not or is only slightly involved, i.e., when $p_{l,i} = 0$ and $p_{l,i} = 0.1$.

7.4.2. Exponentially Distributed Data and Control Plane Service Times

In the last case, the service times of the data plane and control plane devices are exponentially distributed. Again, the approximation error in general for all cases is very small, as shown in Table 3. For the slow and medium controller, the highest errors are caused by increasing the values of $p_{l,i}$ and when there is little controller involvement. For the fast controller, the highest error appears for $p_{,i} = 0.5$ and for increased values of $p_{feed,i}$.

7.5. Evaluation and Comparison of the Approaches

Based on the presented results, the three approximation methods are compared in regards to their approximation precision.

		$p_{feed,i} = 0$	$p_{feed,i} = 0.1$	$p_{feed,i} = 0.5$	$p_{feed,i} = 1$
Slow Controller	$p_{l,i} = 0$	0.5%	5.4%	2.4%	3.0%
	$p_{l,i} = 0.1$	0.3%	5.1%	1.7%	1.3%
	$p_{l,i} = 0.5$	0.3%	6.9%	2.5%	1.7%
Medium Controller	$p_{l,i} = 0$	0.5%	0.7%	2.1%	3.0%
	$p_{l,i} = 0.1$	0.3%	1.1%	1.1%	0.7%
	$p_{l,i} = 0.5$	0.3%	2.4%	1.6%	1.2%
Fast Controller	$p_{l,i} = 0$	0.5%	1.5%	0.3%	0.6%
	$p_{l,i} = 0.1$	0.3%	1.7%	7.0%	4.6%
	$p_{l,i} = 0.5$	0.3%	6.2%	6.7%	9.1%

Table 3: Analytical errors compared to simulation results for three data plane devices and one controller, each with exponentially distributed service times.

7.5.1. Single Data Plane Device

The *M/G/1* approximation approach outperforms the other two approaches in almost every case significantly. Moreover, the errors for Diffusion and Modified Diffusion approach increase significantly if a distribution with a coefficient of variation $c_V > 1$ is used for the service times in one or both planes. Using the *M/G/1* approximation, the errors for such cases are much smaller compared to the other approaches. Specifically, the maximum mean error obtained by the *M/G/1* approach for using distributions with coefficients of variation close to 1 or lower for the service times in the P4 forwarding model with a medium or slow controller feedback is 13.9%. Considering a fast controller increases the error up to 18.2%. For a data plane with a service times distribution with coefficient of variation close to 1 and a control plane service time distribution with a coefficient of variation higher than 1, the error goes up to 26.5% for all controller speeds. Considering a data plane whose service times distribution has a $c_V > 1$, the error for a fast controller with any service time distribution is 46.2%. For the medium and slow controller, this error reduces to 18.2%.

7.5.2. Multiple Data Plane Devices

Similar to the single data plane device network, the *M/G/1* approach outperforms the other two approximations significantly in almost all of the considered cases for an SDN with two data plane devices. Only in rare cases, e.g., in a network with two data plane devices with Erlang distributed service times, no controller involvement, and $p_{l,1} = 0.5$, other approaches are better. Moreover, the approximation error for networks consisting of devices with hyperexponential service times is significantly reduced when using the *M/G/1* approach in comparison to the other two approaches. In particular, if all services times in the network follow a distribution with a coefficient of variation close to 1 or lower, the highest error for all considered cases is not higher than 11%. In the considered case, where the service times of the first data plane device follow a distribution with a coefficient of variation higher than 1, the errors do not exceed 11%. This also holds for our considered case with one service times distribution with a coefficient of variation larger than 1 except for rare exceptions such as the fast controller scenario. In that case, the error can go up to 26% for the fast controller scenario.

When considering a network with three data plane devices, where all service time distributions have a coefficient of variation 1 or smaller, the error does not exceed 10% in the worst case.

8. Conclusion

In this paper, we analyzed several approximation approaches (Diffusion, Modified Diffusion, and Independent M/G/1 queues) for the mean sojourn time of a general P4 forwarding device with controller feedback. We did this for the case of a single P4 forwarding device, and for multiple data planes. The data plane and control plane are modeled using queueing theory. We consider three distributions of service times in the data planes and control plane: Erlang, exponential, and hyperexponential distributions to capture both service behaviors with low and high variance. Additionally, the analysis focuses on different regions of controller utilization. We showed that the best approximation for the vast majority of the considered cases is the one in which the queue behavior in the data plane is considered independent from the queue behavior in the control plane. However, there are cases in which either the Diffusion approximation or the Modified Diffusion approximation perform better. This is an advantage of our work as we are able to answer the question which approximation to use depending on the scenario at hand.

As part of our future work, we plan to find exact expressions for the packet mean sojourn time of such general P4 forwarding devices with feedback. Also, analyzing the behavior beyond the first moment and describing it with explicit equations in terms of the distributions is a further objective.

References

[1] N. Kröger, F. Mehmeti, H. Harkous, W. Kellerer, Performance analysis of general P4 forwarding devices with controller feedback, in: Proc. of ACM MSWiM, 2022.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review 38 (2) (2008).

[3] S. Kianpisheh, T. Taleb, A survey on in-network computing: Programmable data plane and technology specific applications, IEEE Communications Surveys & Tutorials 25 (1) (2023) 701–761. doi:10.1109/COMST.2022.3213237.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al., P4: Programming protocol-independent packet processors, ACM SIGCOMM Computer Communication Review 44 (3) (2014).

[5] H. Harkous, M. Jarschel, M. He, R. Pries, W. Kellerer, P8: P4 with predictable packet processing performance, IEEE Transactions on Network and Service Management 18 (3) (2021).

[6] Netronome, Netronome agilio smartnics (2023).
URL `https://www.netronome.com/products/smartnic/overview/`

[7] N. Zilberman, Y. Audzevich, G. A. Covington, A. W. Moore, Netfpga sume: Toward 100 Gbps as research commodity, IEEE Micro 34 (2014).

[8] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, S. Laki, T4P4S: A target-independent compiler for protocol-independent packet processors, in: Proc. of IEEE HPSR, 2018.

[9] H. Harkous, N. Kröger, M. Jarschel, R. Pries, W. Keller, Modeling and performance analysis of p4 programmable devices, in: 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2021, pp. 67–73. doi:10.1109/NFV-SDN53031.2021.9665141.

[10] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, 1st Edition, Cambridge University Press, USA, 2013.

[11] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, P. Tran-Gia, Modeling and performance evaluation of an openflow architecture, in: 2011 23rd International Teletraffic Congress (ITC), IEEE, 2011, pp. 1–7.

[12] K. Mahmood, A. Chilwan, O. N. Sterb, M. Jarschel, On the modeling of OpenFlow-based SDNs: The single node case, Computer Science & Information Technology (2014).

[13] K. Mahmood, A. Chilwan, O. Østerbø, M. Jarschel, Modelling of OpenFlow-based software-defined networks: the multiple node case, IET Networks 4 (5) (2015) 278–284.

[14] Y. Goto, H. Masuyama, B. Ng, W. K. G. Seah, Y. Takahashi, Queueing analysis of Software Defined Network with realistic OpenFlow–based switch model, in: 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016, pp. 301–306.

[15] J. Ansell, W. K. G. Seah, B. Ng, S. Marshall, Making queueing theory more palatable to SDN/OpenFlow-based network practitioners, in: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 1119–1124.

[16] H. T. Dang, H. Wang, T. Jepsen, G. Brebner, C. Kim, J. Rexford, R. Soulé, H. Weatherspoon, Whippersnapper: A p4 language benchmark suite, in: Proceedings of the Symposium on SDN Research, 2017, pp. 95–101.

[17] D. Scholz, H. Stubbe, S. Gallenmüller, G. Carle, Key Properties of Programmable Data Plane Targets, in: Teletraffic Congress (ITC 32), 2020 32nd International, Osaka, Japan, 2020.

[18] M. Helm, H. Stubbe, D. Scholz, B. Jaeger, S. Gallenmüller, N. Deric, E. Goshi, H. Harkous, Z. Zhou, W. Kellerer, G. Carle, Application of network calculus models on programmable device behavior, in: Proc. of ITC, 2021.

[19] N. Kröger, H. Harkous, F. Mehmeti, W. Kellerer, Looking beyond the first moment: Analysis of packet-related distributions in P4 systems with controller feedback, in: Proc. of International Teletraffic Conference (ITC) 34, 2022.

[20] W. Whitt, The queueing network analyzer, The Bell System Technical Journal 62 (9) (1983).

[21] J. M. Harrison, V. Nguyen, The QNET method for two-moment analysis of open queueing networks, Queueing Systems (6) (1990).

[22] B. R. Haverkort, Approximate analysis of networks of PH—PH—1—K queues: Theory & tool support, in: Quantitative Evaluation of Computing and Communication Systems, Springer, 1995.

[23] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Communications Surveys & Tutorials 16 (3) (2014) 1617–1634. doi:10.1109/SURV.2014.012214.00180.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74. doi:10.1145/1355734.1355746.
URL `https://doi.org/10.1145/1355734.1355746`

[25] D. Gross, J. F. Shortle, J. M. Thompson, C. M. Harris, Fundamentals of Queueing Theory, 4th Edition, Wiley-Interscience, 2008.

[26] H. Kobayashi, Application of the diffusion approximation to queuing networks: Part i equilibrium queue distributions, in: Proceedings of the 1973 ACM SIGME Symposium, SIGME '73, Association for Computing Machinery, New York, NY, USA, 1973, p. 54–62. doi:10.1145/800268.809336.
URL `https://doi.org/10.1145/800268.809336`

[27] M. Reiser, H. Kobayashi, Accuracy of the diffusion approximation for some queuing systems, IBM Journal of Research and Development 18 (2) (1974) 110–124. doi:10.1147/rd.182.0110.

[28] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, Wiley-Interscience, USA, 1998.

[29] E. Gelenbe, On approximate computer system models, J. ACM 22 (2) (1975) 261–269. doi:10.1145/321879.321888.
URL `https://doi.org/10.1145/321879.321888`

[30] U. Mitzlaff, Diffusionsapproximationen von warteschlangensystemen, Ph.D. thesis, Technical University Clausthal (1997).

[31] H. Harkous, K. Sherkawi, M. Jarschel, R. Pries, M. He, W. Kellerer, P4RCProbe for evaluating the performance of P4Runtime-based controllers, in: Proc. of IEEE NFV-SDN, 2021.