Modeling and Performance Analysis of P4 Programmable Devices

Hasanin Harkous*, Nicolai Kröger†, Michael Jarschel*, Rastin Pries*, Wolfgang Kellerer†

*Nokia, firstname.lastname@nokia.com
†Technical University of Munich, firstname.lastname@tum.de
Munich, Germany

***Abstract*—In our digitized society, emerging applications require highly performant and flexible networks that can adapt to satisfy varying connectivity needs. P4 as a domain-specific programming language for data plane pipelines introduces the required flexibility through easy-to-use programmability. However, the performance of P4-capable devices is still an open question that has not yet been completely addressed. Understanding whether a P4-enabled device can meet the performance requirements for a specific network function pipeline is key for planning as well as deployment scenarios in a communication provider network.**

In this paper, we propose a simple analytical model that can quickly predict the performance of network functions written in P4 for a given device. The programmable data plane of P4 devices is modeled as a forward queuing system with a variable service rate that depends on the complexity of the configured data path program. On top of the data plane model, the controller's interaction is modeled as a feedback queuing system. We evaluate the accuracy of our model through a parameter study and simulation. The evaluation reveals corner cases, which are analyzed to formalize a constraint on the service rate using model parameters to guarantee stable system performance.

***Index Terms*—P4, Programmable Data Plane, Performance Modeling, Queuing Theory**

I. Introduction

Next-generation networks promise to satisfy stringent networking requirements, in terms of throughput and latency, to enable emerging applications. Satisfying this promise requires upgrading the capabilities of networking devices to be able to adapt to the connectivity needs of new services and applications. The P4 programming language [1] enables the device-independent development of data plane pipelines for network functions. However, when it comes to the deployment and operation of these network functions, the performance of the P4 target devices is an important factor. While devices sharing the same target architecture are capable of executing the data plane program, there may be significant differences in processing power and forwarding performance depending on the hardware implementation. Therefore, it is key to understand whether a specific device can meet the performance requirements in conjunction with a specific P4-based network function and vice versa. This is true for network planning, dimensioning, and purchasing considerations on the macro-scale but also for inter- and intra-site function placement decisions within an already operational network infrastructure.

Since not all combinations of target devices and P4-pipelines, as well as load scenarios, can be measured and tested, we propose the use of an analytical model to provide quick feedback on the expected performance of a P4 pipeline on a specific device. For this purpose, we create a simple model of P4 programmable devices that can be parameterized to match the setup in question and predict the packet's mean sojourn time. Furthermore, we conduct a parameter study and verify the model through simulation, which helps us to derive a performance constraint for the stable operation of the investigated devices.

The remainder of this paper is structured as follows. Section II revises related work. In Section III, the system's model is proposed and illustrated. In Section IV, the model is extensively evaluated and validated. Finally, the work is concluded in Section V, and future research directions are highlighted.

II. Related Work

Modeling the performance of packet processors is important for planning, predicting, and mitigating problems in networks.

Jarschel et al. [2] analyze an OpenFlow-based architecture where the switch and the control plane are each modeled as M/M/1 queues. The previous model is then mapped to a general Jackson model by Mahmood et al. [3] to be used in modeling OpenFlow networks as Jackson networks in [4]. Goto et al. [5] refines the model of [2] for one node by including processing priorities for packets going to the switch. For practical usage, a control plane application is introduced by Ansell et al. [6] for monitoring networks and predicting the behavior based on queuing theory.

While literature is rich with papers analyzing and modeling the performance of OpenFlow-based switches, few works target modeling the performance of devices with programmable data planes. Dang et al. [7] evaluate the latency performance of three software and emulated P4 devices, i.e. BMv2, PISCES, and P4FPGA when running legacy P4-14 constructs. Scholz et al. [8] measure and model the resource utilization of the ASIC-based Tofino switch, while latency and throughput metrics were selected in modeling the performance of the T4P4S software switch.

None of the previous works consider deriving a complete analytical model for the performance of P4 devices. In [9],

a network calculus-based model is proposed for analyzing the worst-case data plane performance of OF and P4-based switches. While network calculus-based models focus on the worst-case analysis, in this work we consider stochastic models based on queuing theory that can provide information about the mean of different network metrics. Moreover, this work incorporates the controller's impact on the overall performance of the system.

In [10], we benchmarked the packet processing latency of a Netronome SmartNIC when running different P4 constructs. The results are then used to build a simple model for predicting the packet processing latency when running full P4 programs on the SmartNIC. This work was extended in [11] where packet processing latency of the NetFPGA-SUME and the T4P4S software switch beside the Netronome SmartNIC are benchmarked when running a wider set of atomic P4 constructs. Apart from that, a more accurate estimation method is derived based on these benchmarking results to perform a priori predictions regarding the packet processing latency when running arbitrary P4 programs on any of the benchmarked devices. The impact of processing a scaled number of flows on packet processing latency was then investigated in [12].

The model proposed in this paper builds on top of the measurements conducted in [11] and extends it to derive an analytical model for characterizing a P4 device's behavior when considering variable input traffic intensities as well as the impact of different levels of controller's interaction.

III. P4 Queuing Model

In this section, we introduce our queuing model for P4. First, we list the desired requirements to be satisfied by the model. Then, we describe a packet's life cycle when traversing P4 devices. Based on the latter, we propose the model focusing on the data plane and controller components. Finally, we revisit our assumptions.

A. Model Requirements

The model should be capable of capturing the performance of P4 programmable devices while satisfying the following requirements:

- The model takes into account that the packet processing taking place at the P4 data plane varies based on the configured P4 program/pipeline. This affects the packet processing latency, even on the same device, when packets traverse pipelines with different complexities.
- The model takes into account the variable levels of SDN Controller involvement in processing packets that have no matching rule at the data plane; typically, these are the first packets of every newly arriving flow.
- The model takes into account that a variable input traffic load, which depends on the use case scenario, needs to be processed by the P4 device.
- The model should be simple to enable a first-hand understanding of the device's behavior and limitations as well as quick computation.

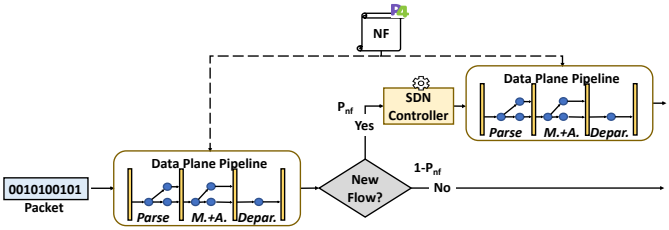


Figure 1: Packet's Life cycle in P4 programmable device.

B. A Packet's Life Cycle in a P4 Device

A P4 program describes the data plane/pipeline of the device that packets will transit. This pipeline describes the logical flow of defined match-action units. First, headers are extracted from the packet at the parser stage. Then, the packet passes through a control flow of match-action units applied on the extracted headers. The match-action unit is the basic unit for packet processing in P4, which includes a table with matching keys along with a list of possible custom actions that can be invoked upon matching. Finally, the packet goes through the deparser stage, where the modified header stack is added again to the packet before it leaves through the designated egress port.

The life cycle of a packet going through a P4 programmable device is shown in Fig. 1. When a packet arrives at the switch from an ingress port, it is processed according to the configured P4 data plane pipeline. If the packet matches any rule installed in the match-action unit, it is processed according to that rule. Otherwise, the packet is forwarded to the controller, which takes the packet forwarding decision based on the running control plane applications. The processed packet is sent back to the data plane along with the forwarding rule that is installed into the device's match-action units. Following packets that match the installed rule will be processed only at the data plane level. Thus, a packet can either be processed at the data plane once or twice in case it is forwarded to the controller.

The analysis focuses on the packet forwarding latency (sojourn time) throughout the system as the key performance metric as well as the dropping probability. The system's sojourn time is calculated according to Eq. 1. It is equal to the summation of the sojourn time of the two paths explained before: *(i)* data plane sojourn time, denoted as E_d; *(ii)* the controller sojourn time, denoted as E_c, along with the data plane sojourn time only when a packet corresponding to a new flow is forwarded to the controller with probability P_{nf}.

$$E_{sys} = E_d + P_{nf} * (E_d + E_c) \tag{1}$$

C. A Simple Model for the System

To capture this forwarding behavior, we propose to model the P4 programmable devices, similar to [2], as a feedback-oriented queuing system. The data plane forwarding is abstracted as an *M/M/1* queuing system and the controller as a feedback queuing system of the type *M/M/1/S*. Fig. 2 shows the overall proposed model, where the external arrival process

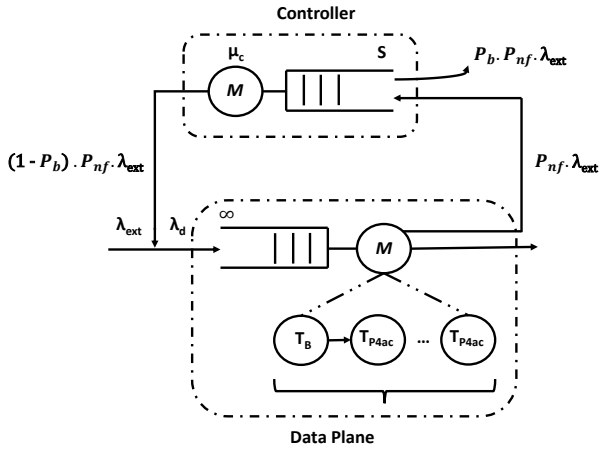


Figure 2: A simple model for P4 programmable devices.

to the switch is assumed to be a Poisson process with a rate equal to λ_{ext}. A Poisson arrival process is selected as it is a convenient mathematical representation of input traffic in many communication systems. As we are working with time averages, the PASTA property of the Poisson process allows us to assume the arrival times as Markovian [13]. In the following, we elaborate on each of the data plane and controller models.

1) Data Plane: The packet processing taking place on a P4 device varies based on the configured P4 program loaded into the device, where more complex network functions result in longer processing delays. This imposes a challenge in capturing the varying performance of the data plane as it can arbitrarily vary based on the complexity of the network functionality described in the configured P4 program. To address this challenge, we leverage the fact that P4 programs are made up of a limited set of atomic P4 constructs/operations. When these P4 constructs are combined in a P4 program, they can describe arbitrary network functions. For example, the Layer 3 Forwarding network function is made up of five P4 constructs: *(i)* Two parse header operations (for Ethernet and IPv4); *(ii)* One table matching on IPv4 destination address; *(iii)* Two header modification operations (for Ethernet and IPv4).

In a previous work [11], we benchmarked three different P4 programmable devices reporting the average latency cost of processing different atomic P4 constructs. We showcased that the average packet processing latency of a P4 device is the summation of two terms: *(i)* the average base latency of a device which captures the processing delay when a packet traverses the non-P4 programmable blocks in a device; *(ii)* the latency due to the processing defined in the P4 program, which is equal to the summation of the average latency cost of all P4 constructs constituting the desired network functionality when running on a P4 device.

Building on these findings, we model the data plane of a P4 device as a queuing system made up of a queue followed by a series of sequential servers. The first server represents the base processing that takes place in a P4 device, and the following servers represent the processing of every P4 construct defined in the loaded P4 program. To simplify the model, we assume that all service processes, when combined, form a service process with exponentially distributed service times with an average service rate equal to $\mu_d^D(NF)$ calculated as shown in Eq. 2.

$$\mu_d^D(NF) = (T_B^D + \sum_{P4ac} T_{P4ac}^D)^{-1} \quad \forall P4ac \in NF, \forall D \quad (2)$$

where $\mu_d^D(NF)$ is the data plane service rate when running network function NF on device D. The terms T_B^D and T_{P4ac}^D refer to the average service times of the base packet processing and the processing of the atomic P4 constructs $P4ac$ on a device D respectively. The service rate of the data plane is calculated as the inverse of the summation of the service times of the base processing component and the component due to P4 constructs processing.

The queue size of the data plane model is assumed to be infinite. The arrival process into the data plane has rate λ_d, evaluated as shown in Eq.3, which is equal to the combination of the externally arriving packets with rate λ_{ext} and the loop-backed packets from the controller [3].

$$\lambda_d = \lambda_{ext} + (1 - P_b) * P_{nf} * \lambda_{ext} \quad (3)$$

where P_b is equal to the dropping probability at the controller, which is assumed to have a limited buffer capacity.

2) Controller: The controller is abstracted as a feedback queuing system on top of the data plane model. The service times of the controller are assumed to be exponentially distributed to keep the model simple, and the queue capacity in the system to be limited to buffer only S packets similar to the approach taken in [2]. As the data plane model is assumed to be an *M/M/1* model with an exponentially distributed service times, the departure process leaving the data plane will follow a Poisson process. Recalling that packets corresponding to new flows will be forwarded to the controller only once after being processed at the data plane, the arrival process incoming to the controller will then be a Poisson process with an arrival rate equal to $P_{nf} * \lambda_{ext}$.

This results in an *M/M/1/S* system for the overall controller system. As the controller has a finite buffer capacity, there is a possibility, with probability P_b, that packets get dropped while waiting to be processed at the controller. Accordingly, only a packet rate of $(1 - P_b) * P_{nf} * \lambda_{ext}$ leaves the controller back to the data plane as shown in Fig. 2.

The model under consideration resembles a Jackson network [13]. Therefore, the average sojourn time of packets traversing the system, shown in Eq. 4, can be calculated by substituting the sojourn time equations of the *M/M/1* system at the data plane and controller in Eq. 1. Note that $\mu_{P4ac}^D(NF)$ and λ_d can be calculated as illustrated in Eqs. 2 and 3 respectively, while μ_c and λ_c correspond to the controller's service and arrival rates respectively.

$$E_{sys} = \frac{1}{\mu_d^D(NF) - \lambda_d} + P_{nf} * (\frac{1}{\mu_d^D(NF) - \lambda_d} + \frac{1}{\mu_c - \lambda_c}) \quad (4)$$

D. Assumptions

To simplify the model, it is assumed that the external arrival process coming to the switch follows a Poisson distribution.

Additionally, exponentially distributed service times are assumed for all servers in the system. Moreover, a single queue is used to abstract incoming traffic over different ports of the switch. The model works well with TCP traffic, where the first packet of every newly observed flow is forwarded to the controller. Finally, the analysis and evaluation conducted in this work focus on steady-state average results of the system rather than distributions.

IV. Model Evaluation

In this section, we evaluate the derived model under different scenarios and validate the results using simulations. Finally, we highlight the performance bottlenecks revealed by the model and the corresponding limits.

Measurements conducted in our previous work [11] are used to characterize the forwarding latency of P4 devices when running different P4 programs. In [11] the forwarding latency of three P4-based network functions is estimated by adding the latency cost of their constituting atomic P4 constructs. The estimations were then validated against the real measured latency when running these three programs on three different P4 devices: *(i)* An Agilio CX 2x10GbE SmartNIC from Netronome, which is a Network Processor Unit (NPU)-based NIC with tens of multi-threaded cores that support an instruction set architecture optimized for packet processing [14]; *(ii)* A NetFPGA-SUME board with Xilinx Virtex-7 XC7V690T FFG1761-3 FPGA [15], which is an FPGA-based packet processor optimized to make use of FPGA programmability while maintaining high packet processing performance; *(iii)* An open-source T4P4S DPDK-based P4 software switch [16], which is a software switch that can run on commodity servers while leveraging the DPDK framework for optimizing packet processing on CPUs. The three programs, which have increasing complexity, are Layer 3 Forwarding (L3FWD), Layer 3 Forwarding with Firewall filtering (L3FWD + Firewall), and VxLAN Decapsulation (VxLAN). Table I summarizes the packet processing latency of the surveyed measurements. These results will be used to derive the service rate $\mu_d^D(NF)$ of the data plane of a P4 device when different P4 programs are loaded.

The probability of a new flow, P_{nf}, is varied from zero, where all packet processing takes place on the data plane, to one where the controller handles every packet arriving at the switch. The latter can be useful in an experiment testing new southbound protocols where every packet needs to be forwarded to the controller, such as the case in [17]. Values of 0.2 and 0.5 are selected as intermediate values, while $P_{nf} = 0.04$ is selected as it presents the probability that a switch will observe a new flow in a normal productive network carrying end-user traffic according to [18]. We evaluate the model based on three different controller's service times: 31μs, 240 μs, and 10000 μs to accommodate for a wide range of possible controller's performance [2]. The largest controller's service time is based on the findings of [19] on the performance of P4Runtime based controllers. The buffer size of the controller system is selected to be $S = 512Bytes$ as a middle ground between experimental and commercial switches.

Table I: Packet Processing latency measurements of three evaluated network functions on three P4 targets [11].

NF	P4 target	Measured Avg. Latency
L3FWD	T4P4S	45.9 μs
	Netro. SmartNIC	8.2 μs
	NetFPGA-SUME	3.7 μs
L3FWD + Firewall	T4P4S	45.9 μs
	Netro. SmartNIC	8.9 μs
	NetFPGA-SUME	4.0 μs
VxLAN	T4P4S	45.9 μs
	Netro. SmartNIC	15.2 μs
	NetFPGA-SUME	5.2 μs

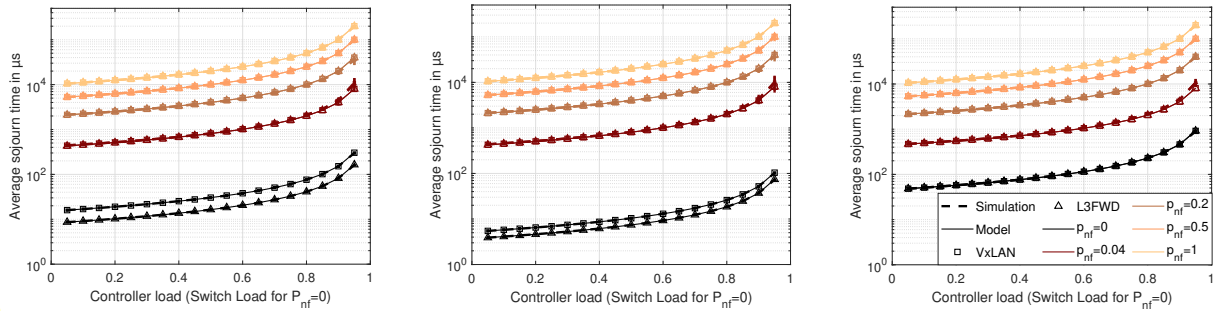
The model presented in the previous section, shown in Fig. 2, is validated using simulations. We chose to validate the model by simulations rather than measurements because simulation results can be obtained faster while still being able to widely vary different parameters of the model. To this end, we implemented a packet-based simulation using MATLAB [20] to reflect the adopted model. To adhere to the realistic behavior of the packet's life cycle, the simulation ensures that a packet may visit the controller at most once. In the simulation, the service times of the controller and the data plane are set to be exponentially distributed, while the external arrival process is set to follow a Poisson process likewise the analytical model. The Poisson assumption on the arrival process to the controller, which is the departure process from the data plane, as well as the one on the arrival process to the data plane from the controller, which is the loopback traffic, are relaxed. This way, simulations can reflect the real interaction between the controller and data plane systems.

Each simulation presented in this section with different parameter sets is performed on around 1.3 million packets and repeated 5 times with different seeds. The 95% confidence interval is plotted on top of the average simulation results. Note that the drop probability at the controller, P_b, is evaluated and always found to be almost equal to zero.

The sojourn time, evaluated based on Eq. 4, derived from the proposed model along with the simulation results are presented in Subsections IV-A, IV-B, and IV-C when the controller service time is set to 10ms, 240μs, and 31μs respectively. In Subsection IV-D, we make use of the derived results to provide a constraint for properly dimensioning the usage of systems with P4 devices.

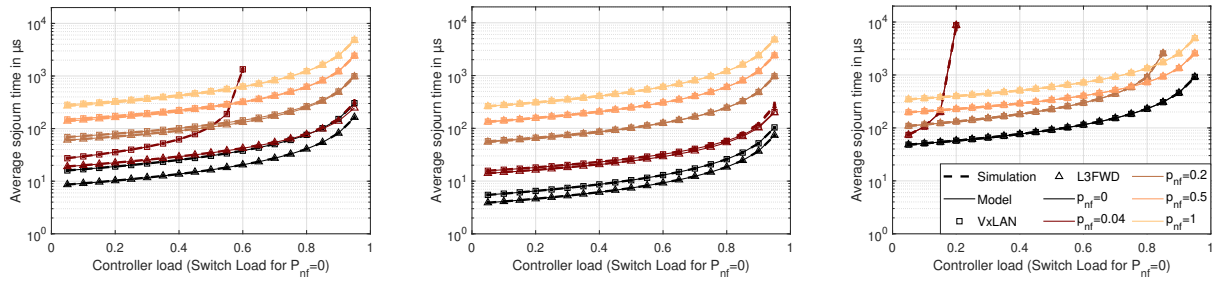
A. Slow Controller Case

Fig. 3 shows the analytical and simulation results of the average sojourn time in μs and in logarithmic scale, as a function of controller load, ρ_c, for different P_{nf} values when controller service time is set to 10 ms. The results shown in Figs. 3a, 3b, and 3c correspond to the cases when Layer 3 Forwarding and VxLAN Decapsulation P4 pipelines are evaluated on Netronome SmartNIC, NetFPGA-SUME, and T4P4S software switch respectively, where data plane service times are taken from Table I. Note that the results corresponding to the L3FWD + Firewall pipeline are not plotted because they



(a) Netronome SmartNIC. (b) NetFPGA-SUME. (c) T4P4S Switch.

Figure 3: Average sojourn time of the system when Controller's service time set equal to 10 ms.



(a) Netronome SmartNIC. (b) NetFPGA-SUME. (c) T4P4S Switch.

Figure 4: Average sojourn time of the system when Controller's service time set equal to 240 μs.

are similar to L3FWD results, since their service times are similar, and they are consistent with the observed patterns. Also note that when $P_{nf} = 0$, the x-axis varies according to the data plane load as the controller's load is always equal to zero.

In all these results, we can observe that the sojourn time increases almost linearly up to a load value equal to 0.8, where the latency increases more sharply when the load approaches full utilization, i.e., equal to one. Moreover, we can also observe that the derived analytical equations always capture the simulated system's behavior under various loads.

Looking at the case where $P_{nf} = 0$, the system's performance is only impacted by the data plane performance. In this case, we can observe that the increase in the data plane's service time has an offset impact, where the curves are shifted upward when the service time of the data plane increases as in the cases of NetFPGA-SUME followed by Netronome SmartNIC and finally T4P4S software switch in Figs. 3b, 3a, and 3c respectively. Additionally, when the load is small, the sojourn time is almost equal to the service time of the data plane. For example, as shown in Fig. 3a, the sojourn time is respectively equal to 8.6 and 16 μs in cases of Layer3-Forwarding and VxLAN-Decapsulation when running on the Netronome SmartNIC with a load equal to 0.05.

The sojourn time shifts upward increasing 2 to 3 orders of magnitudes when the P_{nf} values increase from zero to 1. This high increase in latency is due to the controller's involvement in packet processing who has a very high service time, i.e., 10ms, compared to the data plane's service time. Moreover, we can also observe that the impact of loading different P4 pipelines becomes negligible when P_{nf} values increase where the small difference in the data plane's service time is obscured by the controller's long service time.

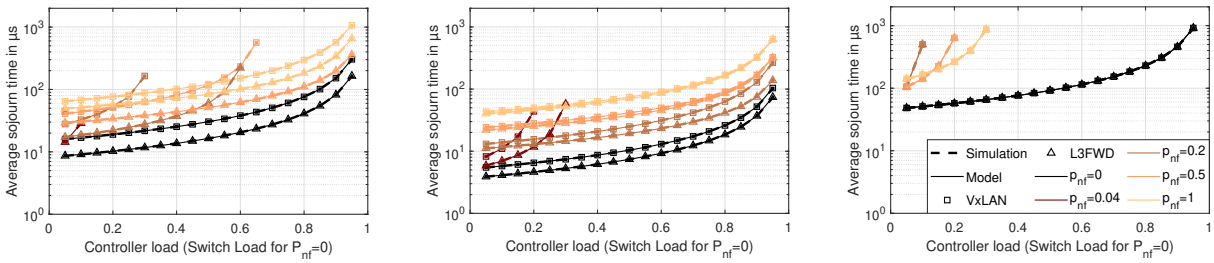
B. Average Controller Case

In this subsection, we evaluate the differences in the system's performance when the controller's service time is smaller, i.e. relatively faster than the case in Subsection. IV-A. Corresponding to Fig. 3, Fig. 4 shows the analytical and simulation results of the average sojourn time for different cases when the controller's service time is set to 240 μs.

Note that as most of the previously analyzed observations still hold, we will only analyze the changes in performance taking place due to the decreased controller's service time. In this case, the data plane performance is playing a larger role in determining the system's performance as the controller's service time decreased. Looking at the cases when P_{nf} is low, the performance curves corresponding to different P4 pipelines are more distinct especially in Fig. 3a, where the difference in pipelines' service time is larger in the Netronome card case compared to other devices.

When P_{nf} increases, the sojourn time increases around one to two orders of magnitude. This smaller impact of P_{nf} on the sojourn time is due to the smaller service time needed by the controller to process packets corresponding to newly observed flows forwarded to it.

Interestingly, we observe that there is a sharp increase in latency occurring in some of the cases breaking the previously



(a) Netronome SmartNIC. (b) NetFPGA-SUME. (c) T4P4S Switch.

Figure 5: Average sojourn time of the system when Controller's service time set equal to 31 μs.

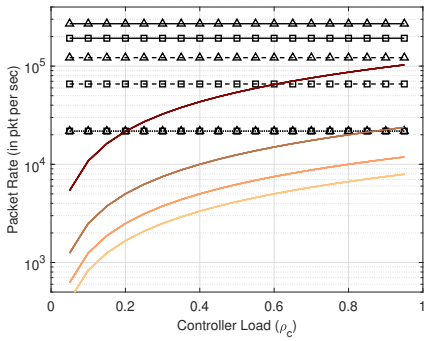
observed patterns. This sharp increase is occurring in the following cases: *(i)* Netronome SmartNIC at $P_{nf} = 0.04$ with VxLAN pipeline, *(ii)* T4P4S software switch at $P_{nf} = 0.04$ and at a later stage at $P_{nf} = 0.2$ with L3FWD and VxLAN pipelines. Note that in these cases, we skipped plotting the sojourn time after some point as the values grow arbitrarily large. The reason behind this sudden increase in latency is that the load/utilization of the data plane in these cases is approaching one. More importantly, we can observe that the analytical model can still capture the system's performance when approaching these corner cases. This issue will never occur for the controller system as we are explicitly varying it between 0.05 and 0.95 in all tested cases. Detailed analysis on this behavior is provided in Subsection IV-D.

C. Fast Controller Case

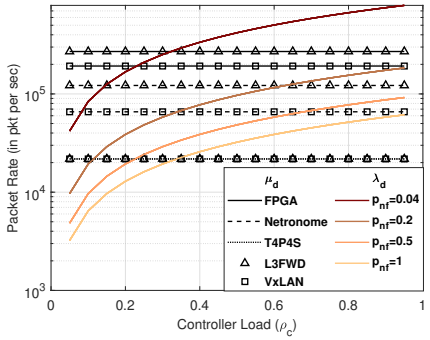
Fig. 5 shows the average sojourn time of the different considered cases when the controller is fast with an average service time equal to 31μs. In this case, similar to the case in Subsection IV-B, the data plane plays a larger role in determining the system performance when latency induced by the control plane processing is reduced. In general, we can see that the overall latency is decreased compared to the latter two cases, and the impact of loading different P4 programs is amplified.

Moreover, we can clearly observe that the limit of data plane utilization is often violated. This is observed in each of the following cases: *(i)* In Netronome SmartNIC, the violation occurs in both pipelines when $P_{nf} = 0.04$ and 0.2, and only in VxLAN case when $P_{nf} = 0.5$, *(ii)* In NetFPGA-SUME, the violation occurs in both pipelines when $P_{nf} = 0.04$, *(iii)* In T4P4S software switch, the violation occurs in all cases except when $P_{nf} = 0$ when the data plane's load is explicitly configured. This violation occurs at different ρ_c values, and when the violation happens at an early stage before $\rho_c = 0.05$, the curve is not shown at all as in the case observed in Fig. 5c when $P_{nf} = 0.04$.

Note that in all cases, the confidence intervals are very small. It is barely visible in high load cases where the system performance approaches instability. Additionally, the deviation of the model results from simulations is evaluated in all cases and found to be on average less than 1.5% for all controller service rates.



(a) Controller's service time set to 240 μs.



(b) Controller's service time set to 31 μs.

Figure 6: Data plane's service rate and arrival rate for different cases.

D. Derived Performance Constraint

In the following, we derive a constraint for dimensioning the system to avoid packet drop. In the previous evaluation, we observed that the sojourn time increased arbitrarily in some of the cases. The reason is that the data plane system is over-utilized. Note that the controller's load never exceeds one because we explicitly vary it between 0.05 and 0.95. To keep the data plane system stable, the load should strictly be less than 1. In other words, the packets arrival rate to the data plane should be less than the service rate of the data plane. Expressing the external arrival rate as a function of the controller load as $\lambda_{ext} = (\mu_c * \rho_c)/P_{nf}$, and substituting it in Eq. 3 of the data plane arrival rate, we get the following constraint:

$$\mu_d^D(NF) > \frac{\mu_c * \rho_c * (1 + (1 - P_b) * P_{nf})}{P_{nf}} \quad (5)$$

Given the anticipated traffic characteristics for a use case scenario, this constraint can help select the appropriate P4 device for running the intended network functionality based on its service rate or inversely its forwarding latency. Alternatively, the constraint can be used to dimension the permissible volume of incoming external traffic before packet drop takes place on a given P4 device with a specific service rate. In Fig. 6, we evaluate and plot, in logarithmic scale, the arrival rate at the data plane (right hand term of Eq. 5) as a function of ρ_c for different P_{nf} values. Additionally, the service rates corresponding to the different P4 pipelines and P4 devices, i.e., equal to the inverse of the previously provided service times, are plotted in this figure. Figs. 6a and 6b correspond to the cases when controller service rates are set equal to (240 μs)$^{-1}$ and (31 μs)$^{-1}$ respectively. Note that the case with the controller's service time equal to 10ms is not shown as the data plane utilization is always stable. The cases when $P_{nf} = 0$ are skipped because the data plane utilization is explicitly set to less than 1 in this case. As long as the service rate is greater than the arrival rate with different P_{nf} values, the data plane system and thus the overall system is in a stable state.

By looking at Figures 6a and 6b, it can be observed that the constraint presented in Eq. 5 is violated at P_{nf} and ρ_c values that match the cases where the sojourn time, shown in Figs. 4 and 5 respectively, increases arbitrarily and is thus not plotted. For example, let us consider the case where the controller service time is equal to 240 μs in Fig. 6a. It can be seen that the arrival rate corresponding to $P_{nf} = 0.5$ and 1 never exceeds any of the data plane service rates. On the contrary, the arrival rate corresponding to the case when $P_{nf} = 0.04$ crosses the service rate of the T4P4S switch for both pipelines after $\rho_c = 0.2$ and crosses the service rate corresponding to running the VxLAN pipeline on the Netronome SmartNIC after $\rho_c = 0.6$. The same applies for the case when $P_{nf} = 0.2$, where it only crosses the service rate corresponding to the T4P4S switch after $\rho_c = 0.85$. The same analysis can be applied by cross-checking the findings in Subsection IV-C with Fig. 6b.

V. CONCLUSION

Deriving analytical models for the performance of networking devices is important for first-hand evaluation and dimensioning of networks. In this work, we abstract the behavior of P4 programmable devices as a forward queuing system at the data plane with variable service rates depending on the complexity of the loaded P4 pipeline. The controller's interaction is modeled as a feedback queuing system on top of the data plane model. The average sojourn service time of the system is evaluated and cross-validated using simulations showing more than 98.5% accuracy. Results show that increasing the data plane service time has an offset effect on the average sojourn time of the system. Crucially, the evaluation revealed corner cases in the system performance. The latter were analyzed to derive a constraint on devices' service rate for guaranteeing a stable system's behavior.

In future work, the model can be further refined by considering a more generic service process based on the distribution of the measured packet processing latency. Moreover, a polling system can be used to abstract the behavior of switches with multiple ingress lines.

REFERENCES

[1] 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*, vol. 44, no. 3, pp. 87–95, 2014.

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

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

[4] K. Mahmood, A. Chilwan, O. Østerbø, and M. Jarschel, "Modelling of OpenFlow-based software-defined networks: the multiple node case," *IET Networks*, vol. 4, no. 5, pp. 278–284, 2015.

[5] Y. Goto, H. Masuyama, B. Ng, W. K. G. Seah, and 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.

[6] J. Ansell, W. K. G. Seah, B. Ng, and 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.

[7] H. T. Dang, H. Wang, T. Jepsen, G. Brebner, C. Kim, J. Rexford, R. Soulé, and H. Weatherspoon, "Whippersnapper: A P4 language benchmark suite," in *Proceedings of the Symposium on SDN Research*, 2017, pp. 95–101.

[8] D. Scholz, H. Stubbe, S. Gallenmüller, and G. Carle, "Key properties of programmable data plane targets," in *2020 32nd International Teletraffic Congress (ITC 32)*, 2020, pp. 114–122.

[9] M. Helm, H. Stubbe, D. Scholz, B. Jaeger, S. Gallenmüller, N. Deric, E. Goshi, H. Harkous, Z. Zhou, W. Kellerer, and G. Carle, "Application of network calculus models on programmable device behavior," in *2021 33rd International Teletraffic Congress (ITC 33)*, Avignon, France, 2021.

[10] H. Harkous, M. Jarschel, M. He, R. Priest, and W. Kellerer, "Towards understanding the performance of P4 programmable hardware," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2019, pp. 1–6.

[11] H. Harkous, M. Jarschel, M. He, R. Pries, and W. Kellerer, "P8: P4 with predictable packet processing performance," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2846–2859, 2021.

[12] H. Harkous, M. He, M. Jarschel, R. Pries, E. Mansour, and W. Kellerer, "Performance study of P4 programmable devices: Flow scalability and rule update responsiveness," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[13] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[14] Netronome. Netronome smartnic. https://www.netronome.com/products/smartnic/overview/. Accessed: 2021-07-04.

[15] Xilinix. (2018) Virtex-7 FPGAs. https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html. Accessed: 2020-01-25.

[16] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, and S. Laki, "T4P4S: A target-independent compiler for protocol-independent packet processors," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2018, pp. 1–8.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," vol. 38, no. 2. Association for Computing Machinery, 2008, p. 69–74.

[18] F. Wamser, R. Pries, D. Staehle, K. Heck, and P. Tran-Gia, "Traffic characterization of a residential wireless internet access," *Special Issue of the Telecommunication Systems (TS) Journal*, vol. 48: 1-2, 2010.

[19] H. Harkous, K. Sherkawi, M. Jarschel, R. Pries, M. He, and W. Kellerer, "P4RCProbe for evaluating the performance of P4Runtime-based controllers," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2021, pp. 1–7.

[20] *MATLAB version (R2020b)*, The Mathworks, Inc., 2020.