Telecommunication Systems
https://doi.org/10.1007/s11235-024-01129-2

Prioritization and offloading in P4 switch integrated with NFV

Farhin Faiza Neha[1] · Yuan-Cheng Lai[2] · Md. Shohrab Hossain[1] · Ying-Dar Lin[3]

Accepted: 2 March 2024

Abstract

The architecture of integrating Software Defined Networking (SDN) with Network Function Virtualization (NFV) is excellent because the former virtualizes the control plane, and the latter virtualizes the data plane. As Programming Protocol-independent Packet Processors (P4) become popular, the architecture integrating SDN with NFV may shift from traditional switches to P4 switches. In this architecture, which integrates P4 switch and NFV (P4 + NFV), network functions can be provided in both P4 switches (PNF) and NFV (VNF). Thus, to minimize packet delay, an offloading problem between P4 switches and NFV in this P4 + NFV should be addressed. This paper tackles this offloading problem and figures out the prioritization mechanism between newly arriving packets and packets that require VNF for minimizing packet delay. We model and analyze the P4 + NFV architecture using an M/M/1 queuing model with non-preemptive priority. Also, we propose an optimization solution based on gradient descent to find the optimal offloading probability of going to VNF. Results show that optimal offloading from P4 switch to NFV can reduce the average packet delay from 13.74 to 40.73%, when packets requiring VNF are given higher priority than newly arriving packets.

Keywords Software defined networking (SDN) · Network function virtualization (NFV) · P4 switch · Priority queue · Optimal probability

1 Introduction

Traditional network architecture is not flexible since hardware devices such as switches and routers cannot easily be updated. Researchers have primarily solved these problems by introducing Software Defined Networking (SDN) where the data plane is decoupled from the control plane [1]. In the control plane, a controller takes all the decisions for finding the data path from the source to the destination. The switches in the data plane only forward the packets according to the decisions determined from the controller. Thus, SDN can be regarded as a technology to virtualize control planes in the network.

On the other hand, the Network Functions (NFs), such as deep packet inspection and load balancer, have been provided by specialized hardware with embedded software. These hardware are usually expensive and cannot easily be updated. Thus, a new technology, Network Function Virtualization (NFV), virtualizes these functions by moving them from relying hardware to being implemented in software installed in off-the-shelf server hardware [2]. NFV virtualizes data plane and creates building blocks that can be linked together to support variety of complex NFs, thereby improving the flexibility and usability significantly [3]. Thus, the architecture of integrating SDN with NFV is an excellent architecture because SDN virtualizes the control plane and NFV virtualizes data plane.

To achieve high-speed forwarding, traditional SDN switches adopt fixed-function chips. However, now some programmable switch chips can process packets as fast as the

✉ Md. Shohrab Hossain
mshohrabhossain@cse.buet.ac.bd

Farhin Faiza Neha
farhinfaiza@gmail.com

Yuan-Cheng Lai
laiyc@cs.ntust.edu.tw

Ying-Dar Lin
ydlin@cs.nctu.edu.tw

1 Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

2 Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

3 Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

fixed-function switches. Thus, P4 (programming protocol-independent packet processors) are proposed. P4 switch uses domain-specific programming language for working in conjunction with SDN controller to achieve protocol-independence, target-independence, and automatic field configuration [4]. In a P4 switch, the programmer can declare the recognized headers of input packets, define the match-action tables and the processing algorithm, and declare the output packets. Moreover, P4 is strong- enough to provide some network functions because it is programmable.

Due to many advantages over traditional switches, P4 switches are more popular and might become mainstream. Thus the hybrid architecture integrating SDN with NFV may be changed from traditional switches to P4 switches, i.e., the integration of P4 switches and NFV (P4 + NFV). In the P4 + NFV architecture, NFs can be provided in both P4 switches and NFV. Therefore, an important issue is how many packets should be forwarded to NFV to obtain NFs (called VNF in this paper) and how many packets should be kept in P4 switch to obtain NFs (called PNF in this paper). Thus, to minimize the packet delay, an offloading problem between P4 switches and NFV in the P4 + NFV architecture should be addressed.

Farhin et al. [5] analyzed the impact of offloading from P4 switch to NFV using multiple VNFs. He et al. [6] first proposed a hybrid architecture by combining P4 switch and NFV for higher flexibility and speed that better suits current network bandwidth requirements. Moreover, Makara et al. [7] analyzed the impact of offloading probability (from P4 switches to NFV) on various performance metrics using Brent's method [8]. Though some recent studies [5–7] analyzed the coordination of P4 switches with NFV, none of the existing works studied the performance gain of this hybrid architecture from the prioritization perspective. Here lies the novelty of our work.

In the P4 + NFV architecture, the packets which require NFs forwarded to NFV will be back to the switch. Recent studies all assumed that these packets and new packets enter the same queue to compete the capacity of processing and communication. In this paper, we separate the packets which return from NFV and new packets into two queues and use a prioritization mechanism to improve data processing efficiency and reduce the average packet delay. Two priority cases: newly coming packets get higher priority (NPH) and packets requiring VNF get higher priority (VPH), are considered. We assess the performance metric using an M/M/1 queuing model with non-preemptive priority analysis in the P4 + NFV hybrid architecture.

Note that the logic behind using the P4 switch as the default data plane is that traffic can be easily offloaded to VNF when the switch is congested. However, an alternative perspective suggests considering the VNF as the default data plane, with traffic offloaded to the P4 switch as needed. This alternative viewpoint argues that packets requiring network functions should be directed to VNF by default, while the P4 switch should only handle traffic when the VNF is inactive. No matter the viewpoint, whether considering VNF as the default or PNF as the default, the approach mentioned in the paper can handle it.

The contributions of the paper are as follows: (i) proposing a P4 + NFV architecture which have different queues for the packets requiring NFV and new packets; (ii) developing an analytical model to analyze this architecture using an M/M/1 queuing model with non-preemptive priority mechanism; (iii) proposing an algorithm based on gradient descent for finding optimal offloading probability from P4 to VNF; and (iv) performing extensive simulations to validate our analytical model and investigate the effects of different parameters on performance metrics.

The rest of the paper is organized as follows. Section 2 describes some previous related works. Section 3 proposes the system model of the P4 + NFV architecture Sect. 4 first analytically derives the average packet delay and then describes an algorithm for finding the optimal offloading probability. Section 5 shows analytical and simulation results to exhibit the performance of different priority settings. Finally, Sect. 6 concludes the paper.

2 Related works

There have been few works on the integration of P4 based programmable switches with NFV. Though some previous studies performed analysis of the SDN/NFV, there has been no previous work that analyzed the performance gain of the P4 + NFV architecture using an M/M/1 queuing model based on priority analysis. In Table 1, we summarize the key points of such previous works according to three categories: *SDN (traditional switch)*, *SDN (traditional switch) + NFV*, and *SDN (P4 switch) + NFV*.

2.1 SDN (traditional switch)

Mahmood et al. [9] proposed a model for OpenFlow based SDN switch, adjusting the traffic arrival rate at one switch. Later, they extended the model for more than one switch in the data plane and analyzed its performance [10]. Nweke et al. [11] worked on a model for finding out consequences of adversarial flow in SDN infrastructure using an M/M/1 queuing model.

Xiong et al. [12] proposed a model for OpenFlow based SDN network and they adjusted batch traffic arrival rate at switch and the packet-in message processing in controller using $M^X/M/1$ and M/G/1 queuing model, respectively. Zhao et al. [13] also improved packet-in message processing in softwarized WAN using an M/G/1 queuing model at a SDN

Table 1 Summary of related works

Category	References	NF queues used?		Characteristics
		VNF	PNF	
SDN (traditional switch)	[9]	No	No	Adjusted traffic arrival rate at single switch using M/M/1 theory
	[10]	No	No	Adjusted traffic arrival rate at multiple switch using M/M/1 theory
	[11]	No	No	Adversarial flow using M/M/1 theory
	[12]	No	No	Batch traffic arrival rate at traditional switch using $M^x/M/1$
	[13]	No	No	Software-defined WAN at traditional SDN switch using M/G/1
	[14]	No	No	Priority based solution with Markov Chain 2D MC (HPQ,LPQ)
	[15]	No	No	Prioritization in 2D MC (HPQ, LPQ) and software vs. hardware switches
	[16]	No	No	Prioritization in 4D MC (internal buffer, HPQ, LPQ, hardware) and encapsulation vs. internal buffer
	[17]	No	No	Priority analysis in SDN (HPQ:M/M/1/k LPQ:M/M/1/k)
	[18]	No	No	Priority analysis and MMPP for multimedia traffic arrivals in SDN
SDN (traditional switch) + NFV	[19]	No	No	Balanced Hash Tree (BHT) for boosting load balancing performance
	[20]	Yes	No	Combination of SDN and NFV using M/M/1
	[21]	No	No	Mobile Cloud Computing using M/M/1 theory
	[22]	No	No	Performance-aware VNF placement algorithm in a hybrid architecture
SDN (P4 switch) + NFV	[6]	Yes	Yes	Fundamental modeling of P4 + NFV architecture
	[23]	Yes	No	5G SDN/NFV Edge with P4 switch
	[24]	Yes	No	NFV framework with event system based on P4 switches
	[25]	Yes	Yes	P4-based Data Plane Programmability and Exposure framework (DPPx)
	[26]	Yes	Yes	MILP based method for boosting capacity in SmartNICs
	[27]	Yes	Yes	Seven state-of-the-art software switches for offloading NFV traffic between NICs and VNFs
	[5]	Yes	Yes	Combination of P4 and NFV using multiple VNFs and M/M/1 queuing theory
	[7]	Yes	Yes	Impact of offloading from P4 switches to NFV using Brent's method
	Our proposed model	Yes	Yes	Combination of P4 and NFV using prioritization mechanism and M/M/1 queuing theory

switch and an M/M/n queuing model at controller to minimize the number of controllers.

Goto et al. [14] proposed a queuing model for OpenFlow based SDN switch which produces very little error. Their model was verified in a test environment.

Singh et al. [15] proposed a model to study tradeoffs between software and hardware switches. They also studied tradeoffs between Encapsulation and Internal buffer in UDP using continuous-time Markov chain [16]. However, in our paper, we argue that software and hardware are not competitors; rather they can be integrated to co-work together.

Some other works [17, 18] used the prioritization mechanism in their analysis. Miao et al. [17] proposed a packet-scheduling technique using an M/M/1 queuing model with non-preemptive priority analysis to reduce packet loss rate in SDN data plane. W. Miao et al. [18] proposed Markov Modulated Poisson Process (MMPP) with priority analysis to model bursty multimedia traffic. The proposed architecture is then used to investigate the network layout and resource share in SDN infrastructure. However, none of the works [9–18] performed analysis for P4 + NFV-based hybrid architecture using the prioritization mechanism.

2.2 Combination of SDN (traditional switch) and NFV based techniques

Lin et al. [19] used the balanced hash tree for decreasing packet processing time and boosting load balancing performance in SDN + NFV architectures. Similarly, Fahmin et al. [20] proposed a hybrid architecture combining SDN and NFV architecture and studied whether NFV should be placed aside or under the controllers. They calculated average packet delay considering M/M/1 queueing theory. Billingsley et al. [21] proposed a model for exploring the performance of Mobile Cloud Computing in the presence of SDN + NFV architectures and used M/M/1 queueing theory in their analysis. Zheng et al. [22] developed a hybrid architecture called Hyper by combining programmable hardware and traditional software architecture in NFV where a performance-aware VNF placement algorithm is designed for optimizing resource usage and minimizing packet delay.

2.3 Combination of SDN (P4 switch) and NFV based techniques

Farhin et al. [5] analyzed the impact of offloading from P4 switch to NFV using multiple VNFs. He et al. [6] first proposed a hybrid architecture by combining P4 switch and NFV for higher flexibility and speed that better suits current network bandwidth requirements. Paolucci et al. [23] proposed a method by introducing P4 Data Plane Programmability (DPP) in SDN/NFV for increasing flexibility and facilitate adoption in recent network applications, e.g. 5G networks, IoT, cyber security, and the latest traffic engineering. Ji et al. [24] studied a high performance NFV with an event system in combination with a P4 switch which can enhance the performance gain and provide lower packet delay by supporting multiple function chains at line rate. Osiński et al. [25] used BMv2 (Behavioral Model Version 2) software switch and proposed a P4-based Data Plane Programmability model combining with Exposure framework (DPPx) to escalate flexibility of NFV services. P4 + NFV-based hybrid architecture can also be used for improving the performance of network interface cards (NICs). Mohammad et al. [26] proposed a Mixed Integer Linear Programming (MILP) based optimization method using P4 + NFV architecture for Smart-NICs which can escalate flexibility by reducing packet delay. Zhang et al. [27] studied the performance gain of NFV for offloading traffic between physical NICs and VNFs.

However, no P4 + NFV architecture has been developed for finding out the optimal offloading probability to reduce the packet delay based on priority analysis.

Makara et al. [7] analyzed the impact of offloading probability (from P4 switches to NFV) on various performance metrics using Brent's method [8]. They introduced a controller to decide the route and required operation of a particular packet. However, in our proposed approach, we have analyzed the impact of offloading probability (from P4 switches to NFV) using the prioritization mechanism.

Among all the three categories of previous works, our work is more relevant to the third category of works that analyzed P4 + NFV hybrid architecture. However, none of the works studied the performance benefits of P4 + NFV hybrid architecture using a prioritization mechanism.

3 System model

We first describe the system model that is used for developing the analytical model. Traffic arriving at a switch in need of a NF can be served in the P4 switch (called as PNF) or can be redirected to the NFV in the data center (called as VNF). The packets which are forwarded to VNF will again enter the switch and finally they are directed to their destination. We will use the M/M/1 queuing model based on non-preemptive priority analysis. Moreover, two different priority cases: newly coming packets get higher priority (NPH) and packets requiring VNF get higher priority (VPH), are considered.

3.1 System model

Figure 1 shows the network model of a P4-based programmable switch with a VNF. For calculating average packet delay, we have considered the M/M/1 queuing model using a prioritization mechanism. As shown in Fig. 1, there are six queues in this network model:

(a) *Two switch processing (SP) queues:* Switch processing module processes all packets including the new packets and packets that have visited VNF and reentered the switch. All newly coming packets (shown in black) will enter one queue (denoted as SP1) while packets which have visited VNF (shown in blue) will enter another queue (denoted as SP2).

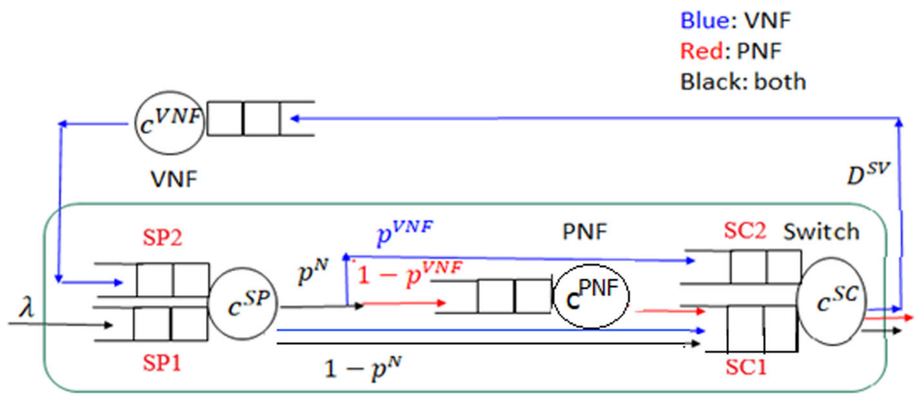

Fig. 1 Queuing model of programmable switch

(b) *A PNF queue:* A PNF module processes packets that require network functions inside the switch (shown in red).

(c) *Two switch communication (SC) queues:* SC module forwards all packets to their next hop. All newly coming packets (shown in black) will enter one queue (denoted as SC1) while packets which require VNF (shown in blue) will enter another queue (denoted as SC2).

(d) *A VNF queue:* Packets that require VNF functions (shown in blue) will queue for the VNF module, which will process these queued packets and subsequently send them back to the switch.

According to Fig. 1, the incoming arrow on the left side indicates the new packet arrivals at a switch. At first, newly coming packets are processed by the P4 switch. Then, the switch processing module determines whether this packet needs NF or not by parsing the packet header. Rules can be defined to match specific patterns in the packet headers, such as source/destination IP addresses, protocol types, or any other fields of interest. Based on the matched rules, decisions are made regarding the application of network functions. Generally, in a typical OpenFlow network, the probability of requiring network function is assumed to be 50%. If it needs NF, the switch processing module further determines whether to serve this packet in the P4 switch itself (PNF) or send it out towards the VNF according to the offloading probability from P4 switch to NFV. After that, the packets which have obtained VNF enter the switch again and finally they are directed to their destinations by the switch's communication module. In our system model, switch processing queues (SP1 and SP2) and switch communication queues (SC1 and SC2) are typically associated with logical ports rather than physical ports. This association is defined in the P4 program and may not directly correspond to physical hardware ports. The programmability of the P4 switch allows for customization and a more versatile design, enabling the mapping between logical ports and physical ports to be influenced by the P4 program and the target device.

In our paper, we implemented a one-way flow without establishing future timeslot relationships to accommodate varying transmission speeds among different devices. Our system involves different processing capabilities, and asynchronous communication proves advantageous for its flexibility. The absence of coordination and scheduling, typically associated with multiple timeslots, makes asynchronous communication suitable for large networks that do not demand tight synchronization. This approach is particularly beneficial in environments with unpredictable communication demands, allowing for variations in traffic patterns.

In our system model, a feedback path is introduced which serves as a way to identify errors or congestion at different processing stages. The switch determines how to act according to the rules, which are set in the controller. Thus, when the packet gets the NF from VNF, it still needs to come back to the original switch to determine which port it should go eventually.

Some assumptions of the model are as follows: (1) Packet arrivals at a switch follow a Poisson process, (2) The size in each queue is infinite, and (3) NF is required only once.

The notations used in the model and later analysis are listed in Table 2.

3.2 Problem statement

In this section, we provide a precise problem statement to guide our analysis and proposed solutions.

Given:

- Switch processing rate: c^{SP};
- Switch Communication rate: c^{SC};
- PNF computing capacity: c^{PNF};
- VNF computing capacity: c^{VNF};
- Packet arrival rate: λ;
- Probability needing network functions: p^N;
- Fixed propagation delay: D^{SV}.

Table 2 Notations used in the analysis

Category	Symbol	Parameter Name
Capacity	c^{SP}	Switch processing rate
	c^{SC}	Switch communication rate
	c^{PNF}	Service rate for PNF
	c^{VNF}	Service rate for VNF
Arrival rate	λ	Packet arrival rate
Probability	p^{N}	Probability of requiring network function
	p^{VNF}	Probability going to VNF
Service rate	ρ^{SC1}	Service rate for queue SC1
	ρ^{SC2}	Service rate for queue SC2
	ρ^{SP1}	Service rate for queue SP1
	ρ^{SP2}	Service rate for queue SP2
Packet delay for three types of packets	d^{PNF}	Packet delay for packets that need network function at PNF
	d^{VNF}	Packet delay packets that need network function at VNF
	d^{ONF}	Packet delay packets that do not need any network function
Average packet delay at queues	t^{SP1}	Average packet delay at SP1 queue
	t^{SP2}	Average packet delay at SP2 queue
	t^{SC1}	Average packet delay at SC1 queue
	t^{SC2}	Average packet delay at SC2 queue
	t^{PNF}	Average packet delay at PNF queue
	t^{VNF}	Average packet delay at VNF queue
Delay	D	Average packet delay for both cases
	D^{SV}	Fixed propagation delay

Output:

- Optimal probability going to VNF: p^{*VNF}.

Objective:

- Minimize average packet delay D.

Constraint:

- $0 \leq p^{*VNF} \leq 1$.

Table 2 also provides a list of notations used in the analysis. In the system model, the SP queue processes incoming packets, while the SC queue forwards packets to their next hop. Therefore, the processing capacities of SP queue and SC queue are denoted as c^{SP} and c^{SC}, respectively. Similarly, processing capacities or service rates of the PNF queue and the VNF queues are denoted as c^{PNF} and c^{VNF}, respectively. The packets arrive at the P4 switch at a rate λ. In a typical OpenFlow network, the probability of requiring network function is assumed to be 50% and is denoted as p^{N}. Additionally, the probability of going to the VNF queue is represented by p^{VNF}. Our goal is to determine the optimal offloading probability to the VNF queue, denoted as p^{*VNF}. For mathematical analysis, three types of packets are considered: packets that require a network function at the PNF queue, packets that require a network function at the VNF queue, and packets that do not require any network function (discussed in Sect. 4). The corresponding packet delays for these types are represented as d^{PNF}, d^{VNF}, and d^{ONF}, respectively. To derive the packet delay for each type, we need to calculate packet delay at the SP queue, SC queue, PNF queue, and VNF queue. Thus, we use the notations t^{SP}, t^{SC}, t^{PNF}, and t^{VNF} to represent the average packet delay at these respective stages. The total average packet delay for the entire architecture is denoted as D. Finally, D^{SV} represents the fixed propagation delay at the i^{th} VNF queue. That is, the problem is to find the optimal p^{VNF}, denoted as p^{*VNF} using a prioritization mechanism to minimize D.

4 Solution

We first derive the formula of average packet delay and then propose an optimization algorithm to find the optimal offloading probability by using the derived formulas.

4.1 Average packet delay

We now calculate the average packet delay of the proposed queuing network for two different priority cases: NPH and VPH. We will use M/M/1 theory considering non-preemptive priority queues.

First, we obtain the packet delay according to the ratios of packets, as

$$D = p^{N}\left(1 - p^{VNF}\right) d^{PNF} + p^{N} p^{VNF} d^{VNF} + \left(1 - p^{N}\right) d^{ONF}. \quad (1)$$

where, d^{PNF}, d^{VNF}, andd^{ONF} represent the delay of packets that require network function at PNF, VNF, and do not require any network function, respectively.

For obtaining, d^{PNF}, d^{VNF}andd^{ONF}, we can sum the delay of their paths. Let t^{SP1}, t^{SP2}, t^{SC1}, t^{SC2}, t^{PNF}, t^{VNF} represent the delay in the queues of SP1, SP2, SC1, SC2, PNF and VNF, respectively.

The packets requiring network function at PNF visits SP and SC queue once, i.e. SP1, SC1 and additionally visit the PNF queue. Thus, its delay is as

$$d^{PNF} = \left(t^{SP1} + t^{PNF} + t^{SC1}\right). \tag{2}$$

The packets requiring network function at VNF visit SP and SC queue twice and VNF queue once (in the sequence of SP1, SC1, VNF, SP2, SC2), causing $2D^{SV}$ propagation delay. Therefore, its delay is determined as

$$d^{VNF} = \left(t^{SP1} + t^{SC1} + t^{VNF} + t^{SP2} + t^{SC2} + 2D^{SV}\right). \tag{3}$$

The packets that do not require any network function only visit SP1 and SC1 queue once, so its delay is as

$$d^{ONF} = \left(t^{SP1} + t^{SC1}\right). \tag{4}$$

Thus, we need to calculate the packet delay in each queue. However, NPH and VPH have different packet delays in the switch's processing queue and communication queue because of different priority settings. Thus, we derive the delay in each queue for NPH and VPH.

Delay in each queue for NPH As shown in Fig. 1, newly coming packets will get higher priority and enter through SP1 queue and then forwarded through SC1 queue.

(i) Delay at SP queue. Primarily packets enter the switch at an arrival rate of λ through the switch's processing queue SP1. Packets which have experienced VNF re-enter the switch using SP2 queue at an arrival rate of $\lambda p^N p^{VNF}$. Thus the load in SP1 and SP2 can be obtained as as $\rho^{SP1} = \lambda/c^{SP}$ and $\rho^{SP2} = (\lambda p^N p^{VNF})/c^{SP}$, respectively. Since we use M/M/1 queuing theory considering non-preemptive priority queue, the average packet delay at the switch's processing queues SP1 and SP2 can be calculated as

$$\begin{aligned} t^{SP1} &= \frac{\left(1+\rho^{SP2}\right)/c^{SP}}{1-\rho^{SP1}}, \\ t^{SP2} &= \frac{\left[1-\rho^{SP1}\left(1-\rho^{SP1}-\rho^{SP2}\right)\right]/c^{SP}}{\left(1-\rho^{SP1}\right)\left(1-\rho^{SP1}-\rho^{SP2}\right)}. \end{aligned} \tag{5}$$

(ii) Delay at SC queue. Packets which does not require network function at VNF enter SC1 queue with an arrival rate λ and packets which require VNF exit through SC2 queue with an arrival rate $\lambda p^N p^{VNF}$. Thus, in NPH priority setting, we have given higher priority to packets that do not require network function at VNF and exit the switch through SC1 queue. Thus, we can obtain the packet delay at the SC1 and the SC2 queue as

$$\begin{aligned} t^{SC1} &= \frac{\left(1+\rho^{SC2}\right)/c^{SC}}{1-\rho^{SC1}}, \\ t^{SC2} &= \frac{\left[1-\rho^{SC1}\left(1-\rho^{SC1}-\rho^{SC2}\right)\right]/c^{SC}}{\left(1-\rho^{SC1}\right)\left(1-\rho^{SC1}-\rho^{SC2}\right)}. \end{aligned} \tag{6}$$

where, $\rho^{SC1} = \lambda/c^{SC}$ and $\rho^{SC2} = (\lambda p^N p^{VNF})/c^{SC}$.

(iii) Delay at PNF queue. The service rate at the PNF queue is c^{PNF} and its arrival rate is $\lambda p^N\left(1-p^{VNF}\right)$. Thus the average packet delay at the switch's PNF queue can be calculated as

$$t^{PNF} = \frac{1}{c^{PNF} - \lambda p^N(1-p^{VNF})}. \tag{7}$$

(iv) Delay at VNF queue. The service rate at the VNF queue is c^{VNF} and its arrival rate is $\lambda p^N p^{VNF}$. Thus average packet delay at VNF is expressed as

$$t^{VNF} = \frac{1}{c^{VNF} - \lambda p^N p^{VNF}}. \tag{8}$$

Delay in Each queue for VPH. In this case, packets which require VNF will get higher priority. After getting VNF service, packets re-enter the switch through the SP2 queue and then leave the switch through the SC2 queue. We need to calculate the average packet delay for the aforementioned three types of packets. Note that here we only show the delay in the queues of SP1, SP2, SC1, and SC2, because they are different from those in NPH due to the different priority settings.

VPH only has different priority settings with NPH, but the load in SP1, SP2, SC1, and SC2 for VPH are the same as that for NPH. Thus we can easily obtain the delay of SP1, SP2, SC1, and SC2 as

$$\begin{aligned} t^{SP1} &= \frac{\left[1-\rho^{SP2}\left(1-\rho^{SP2}-\rho^{SP1}\right)\right]/c^{SP}}{\left(1-\rho^{SP2}\right)\left(1-\rho^{SP2}-\rho^{SP1}\right)}, \\ t^{SP2} &= \frac{\left(1+\rho^{SP1}\right)/c^{SP}}{1-\rho^{SP2}}, \\ t^{SC1} &= \frac{\left[1-\rho^{SC2}\left(1-\rho^{SC2}-\rho^{SC1}\right)\right]/c^{SC}}{\left(1-\rho^{SC2}\right)\left(1-\rho^{SC2}-\rho^{SC1}\right)}, \\ t^{SC2} &= \frac{\left(1+\rho^{SC1}\right)/c^{SC}}{1-\rho^{SC2}}. \end{aligned} \tag{9}$$

4.2 Algorithm for finding optimal offloading probability

Algorithm 1 is known as the P4 switch integrated with NFV (PINOpt) algorithm. PINOpt algorithm is designed to explore the state space and to return the optimal probability (p^{*VNF}) for directing packets to VNF, resulting in the minimum average packet delay. The algorithm employs a searching method based on the gradient descent algorithm.

To begin, we initialize p^{VNF} with the value of p^{INI} (initial probability). Two parameters: *step* and *stepRF*, are used to control the searching range and gradually reduce the step size in each iteration, respectively. Four variables: *minscope, maxscope, optimal, and x,* are initialized. The PINOpt algorithm searches for optimal values of p^{VNF} by generating a search space [*minscope, maxscope*] in each step, shown in line 4 to 5. Moving on to lines 6 to 16, the algorithm finds the optimal p^{VNF} values that correspond to the minimum delay within the defined search space. Initially, the value of *minscope* is copied to the *optimal* variable. The *minDelay* (minimum delay) is initially set to *Infinity* (∞). Then, we assign the value of the *minscope* variable to the value of the *x* variable. The *while* loop continues until the value of the *x* exceeds the value of the *maxscope*. Within the loop, the *DelayforProb(x)* function calculates the average packet delay based on the p^{VNF} value stored in the *x* variable. Equation (1) is used in the *DelayforProb(x)* function to calculate the total average packet delay. The necessary parameter values for the *Delay* (total average packet delay) calculation are listed in Table 3. During the calculation, if the computed *Delay* value is less than the current *minDelay*, *minDelay* is updated accordingly. The optimal p^{VNF} value, which corresponds to the minimum delay, are stored in the *optimal* variable. As we approach the solution, the *x* variable is updated using the *step* value. This *step* value gradually decreases, controlled by the *stepRF* parameter. The required accuracy for finding the optimal p^{VNF} values is controlled by an input parameter ε (precision).

Algorithm 1 PINOpt for finding optimal p^{VNF}

```
Input: $\varepsilon$, $p^{INI}$, $stepRF$
Output: $p^{VNF}$
1  $p^{VNF} = p^{INI}$
2  $step = p^{INI} / stepRF$
3  while ( $step \geq \varepsilon$ ): do
4      $minscope$ = max [0, $p^{VNF}$ - ($step \times stepRF$)]
5      $maxscope$ = min [1, $p^{VNF}$ + ($step \times stepRF$)]

6      $optimal = minscope$
7      $minDelay = \infty$
8      $x = minscope$

9      while $x \leq maxscope$: do
10         $Delay = DelayforProb(x)$ /* Eqn. (1) according to $x$
11         if ($Delay < minDelay$): then
12             $minDelay = Delay$
13             $optimal = x$
14         $x$ += $step$
15     $p^{VNF}$=$optimal$
16     $step = step / stepRF$
17 return $p^{VNF}$
```

Table 3 Baseline parameters for the analysis and simulation

Symbol	Value
c^{SP}	12,500,000 pkts/s
c^{SC}	625,000 pkts/s
c^{PNF}	12,500 pkts/s
λ	125,000 pkts/s
c^{VNF}	95,000 pkts/s
p^{N}	0.5
D^{SV}	10μs
ε	10^{-6}
p^{INI}	0.5
stepRF	10

5 Analytical and simulation results

5.1 Designing a custom simulator

We have designed a custom simulator using Ciw event simulation library for packet generation [28, 29]. Ciw has useful libraries whose core features include the capability to simulate networks of queues, batch arrivals, multiple customer classes, dynamic customer classes, priorities, schedules, and deadlock detection. We have used these features to verify our analytical model. A new packet enters the switch following Poisson distribution. After that, depending on different probability, it goes to the PNF queue, switch's communication queue or the VNF queue. We have created a priority class, packet class and also routing function for packet routing. For executing different queuing events, we have created an event queue where events get generated and are processed sequentially. We have logged each packet's propagation path to obtain the packet delay in different queues. After calculating the average packet delays for both cases (having different priorities as discussed in Sect. 3), we need to find the optimal probability of going to VNF from the P4 switch so that average packet delay is minimized.

5.2 Parameter settings

We have listed all the baseline parameters that are used for the analysis and simulation in Table 3. In a typical OpenFlow network, the probability of requiring network function is 50% and VNF service rate is 95,000 packets/sec [20]. In a typical OpenFlow network, packet arrival rate is 125,000 packets/sec [30]. Similar to some previous work [30, 31], we have kept switch Communication rate as 1 Gbps (625,000 packets/sec) and switch processing rate as 12,500,000 packets/sec. We have run each simulation 100 times and taken the average of each metric.

We first show the impact of p^{VNF} on the average delay to show its significance. Afterwards, we analyze the sensitivity of various parameters, including λ, p^N, c^{VNF}, and D^{SV}.

5.3 Optimal value of p^{VNF}

We have derived the optimal offloading probabilities for both cases i.e. NPH and VPH. For showing results, in the legend we have used NPH(S), NPH(A), VPH(S), and VPH(A). Here, NPH(S) and NPH(A) denote simulation and analytical results, respectively, for new packets with a high priority. On the other hand, VPH(S) and VPH (A) denote simulation and analytical results, respectively, for VNF packets with a high priority.

Figure 2 shows the impact of probability of going to VNF (p^{VNF}) on the average packet delay for both cases. The analytical results match the simulation results, meaning that our

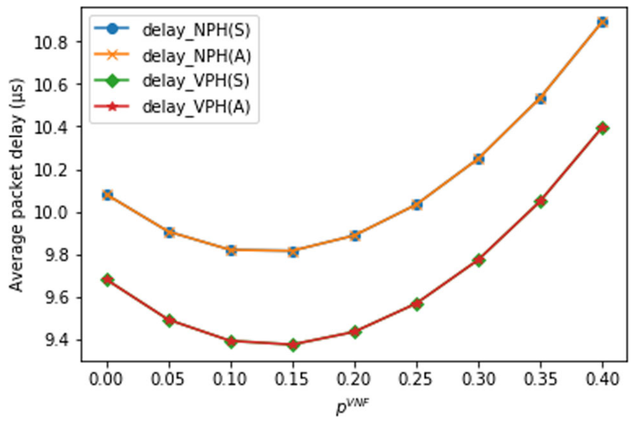

Fig. 2 Impact of p^{VNF} on average packet delay

analytical model can mirror the realistic context precisely. When p^{VNF} is 0 which means no packet is offloaded to VNF, the PNF queue is extremely congested, resulting in greater packet delays. As p^{VNF} increases, the average packet delay decreases because load decreases in the PNF queue, making it less congested. For NPH, average packet delay is lowest (9.81 µs) when p^{VNF} is 0.12781. For VPH, average packet delay is lowest (9.37 µs) when p^{VNF} is 0.13602. However, after a particular point, with the increase of p^{VNF}, average packet delay again tends to increase. This is because more packets go to VNF and the result of the overhead of packets going to the VNF is more intense on the delay.

We can also observe that for NPH, the average packet delay is higher than that of VPH. This is because, in VPH, packets that use VNF get higher priority than PNF. After that, the VNF queue starts building up sooner than the PNF queue, thereby minimizing the average packet delay. In case of VPH, VNF packets with the higher priority will be served first. Therefore, the packets requiring VNF in SP2 and SC2 have shorter delay than the packets requiring PNF in SP1 and SC1, respectively. Also, the load of PNF is heavier than that of VNF due to the extra propagation delay between the switch and VNF. Because of these phenomena, *delay_NPH* is higher than that of *delay_VPH.*

5.4 Impact of packet arrival rate, λ

The impacts of the packet arrival rate λ on the optimal offloading p^{*VNF} and the average packet delay are shown in Figs. 3 and 4, respectively. It is found that the analytical results match the simulation results, meaning that the analytical model can mirror the realistic context precisely.

In Fig. 3, we can observe that with the increase of the λ, p^{*VNF} increases notably for both cases. As λ increases, the packet delay for packets requiring PNF increases more rapidly than the packet delay for packets requiring VNF (as shown in Fig. 5 and will be explained later). This

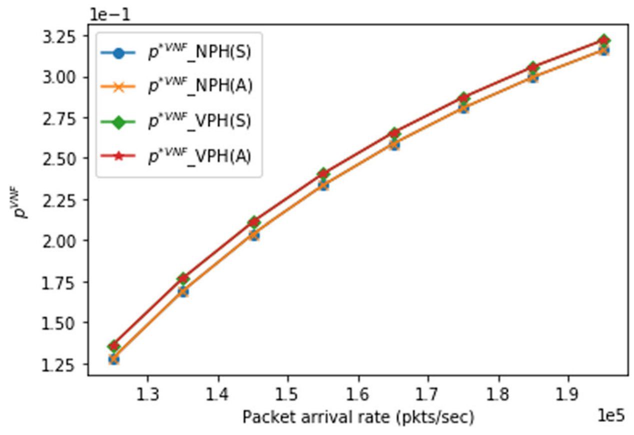

Fig. 3 Impact of arrival rate, λ on probability of going to VNF (p^{VNF})

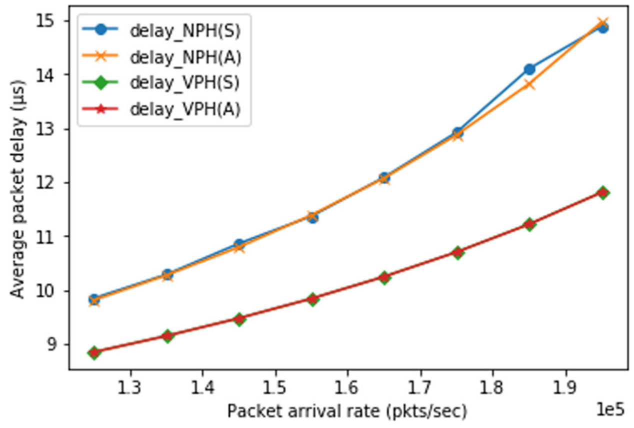

Fig. 4 Impact of arrival rate, λ on average packet delay

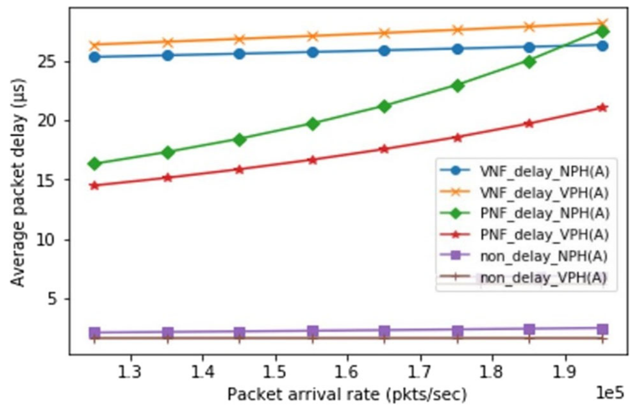

Fig. 5 Impact of arrival rate, λ on packet delay on different paths

phenomenon causes p^{*VNF} to rise and more packets are offloaded to VNF. Conceptually speaking, it is not so effective to offload packets to VNF at a low arrival rate since PNF can handle packets without serious congestion at its queue. However, as load increases, it is appropriate to send more packets to VNF as the PNF queue becomes more congested.

Moreover, after a particular point, with the increase of p^{*VNF}, the slope of the curve starts to decrease. This is because after a particular value of λ, load increases at the VNF queue and it gets congested, thereby diminishing the benefit of offloading.

We can observe that p^{*VNF} in VPH is higher than that of NPH. In VPH, packets requiring VNF get a higher priority than those packets requiring PNF. Therefore, the packets that require VNF experience shorter delay in SP2 and SC2 than the packets requiring PNF in SP1 and SC1, respectively. Thus to minimize the average packet delay, more packets tend to go for VNF service, making p^{*VNF} be larger in VPH than that of NPH.

In Fig. 4, we can observe that with the increase of λ, the average packet delay increases. This is because more packets cause heavier load, resulting in greater packet delays. We can observe that the average packet delay for NPH is higher than that of VPH. This reason has been explained in Fig. 2. With a lower packet arrival rate, the average packet delay for VPH can be reduced by 13.74% (from 9.907 to 8.546 μs, obtained from Fig. 4) as compared to NPH. Moreover, for higher arrival rates, VPH performs even better. At high packet arrival rates, on average, there is a delay reduction of 40.73% (from 17.889 to 10.603 μs, obtained from Fig. 4) for VPH compared to NPH.

In Fig. 5, we further observe the impact of λ on the packet delay of the packets that goes through VNF, PNF and no network function, denoted as *VNF_delay, PNF_delay*, and *non_delay*, respectively. Since the analytical and simulation results match well, we have only used the analytical results.

We can observe that with the increase of λ, the packet delay increases for all packets because more packets cause all queues more congested, resulting in longer delay. However, the increasing slopes of *VNF_delay, PNF_delay*, and *non_delay* are quite different. As the packet arrival rate increases, non_delay is almost stable, *VNF_delay* increases slowly, and *PNF_delay* increases faster. The load of PNF is heavier than that of VNF. This is because the packets offloaded to VNF experience an extra propagation delay between the switch and VNF. For this reason, when packet arrival rate exceeds this point (165,000 pkts/sec), the slope of *PNF_delay* significantly increases due to a heavy load on PNF while the slope of *VNF_delay* only slightly increases due to a light load on VNF.

We can also observe that *PNF_delay* for NPH is higher than that of VPH. On the contrary, *VNF_delay* for NPH is lower than that of VPH. This is because for VPH, the optimal probability p^{*VNF} increases as shown in Fig. 3, and more packets go to VNF than the PNF queue, thereby increasing the load at VNF and decreasing the load at PNF.

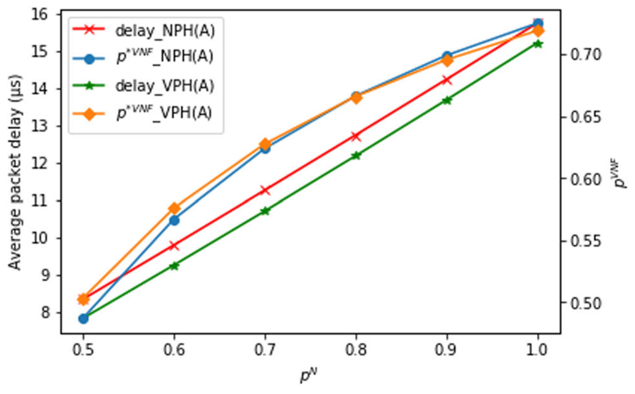

Fig. 6 Impact of p^N on delay and offloading probability

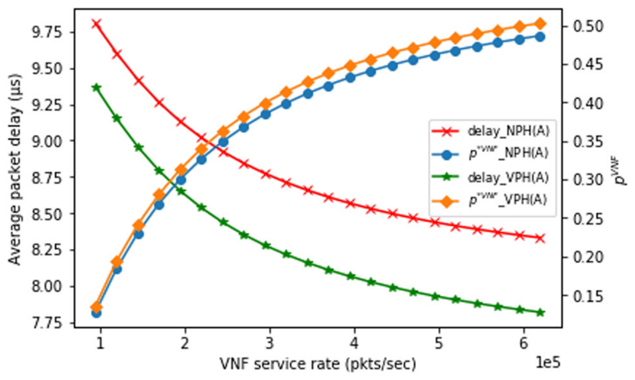

Fig. 7 Impact of c^{VNF} on delay and offloading probability

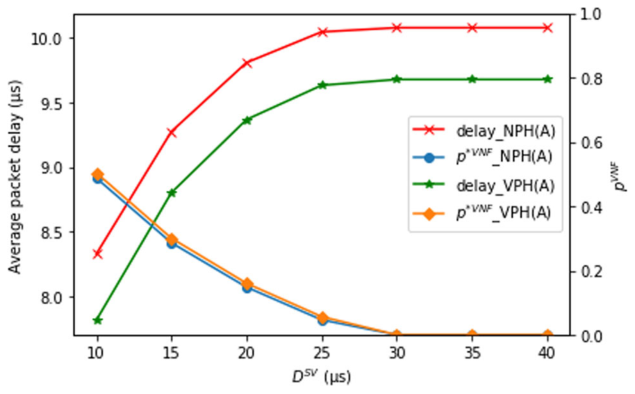

Fig. 8 Impact of D^{SV} on delay and offloading probability

5.5 Impact of p^N

For the next set of results (Figs. 6, 7, 8), the left y axis indicates the average packet delay measured in μs and the right y axis indicates the probability of going to VNF (p^{VNF}). Figure 6 shows the impact of probability of requiring network function (p^N) on the average packet delay and the optimal offloading probability.

Figure 6 shows the impact of probability of requiring network function, p^N, on the average packet delay and the optimal offloading probability. From Fig. 6, we can observe that with the increase of p^N, the average packet delay increases as more packets require NF, increasing the load of PNF and VNF. With the increase of p^N, p^{*VNF} also increases. The load at PNF is heavier than the load at VNF because the PNF service rate (c^{PNF}) is much lower than the VNF service rate (c^{VNF}), as shown in Table 3. Therefore, as p^N increases, the average delay for packets requiring PNF increases more quickly than the average delay for packets requiring VNF, causing that p^{*VNF} increases. That is, offloading more packets to VNF to reduce the average packet delay.

Another observation is that with the increase of p^N, the delay increases linearly whereas p^{*VNF} increases in a logarithmic way. By rearranging the equation of average delay shown in Eq. (1), we get the following linear equation as

$$D = p^N\Big(d^{PNF} - p^{VNF}d^{PNF} + p^{VNF}d^{VNF} - d^{ONF}\Big) + d^{ONF}, \tag{10}$$

which generates a linear increase in delay.

When p^N is small, p^{*VNF} in VPH is larger than that of NPH. In VPH, the packets that require VNF experience shorter delay in SP2 and SC2 than the packets requiring PNF in SP1 and SC1, respectively. Thus to minimize the average packet delay, more packets tend to go for VNF service, making p^{*VNF} be larger in VPH than that in NPH. However, when p^N is large, excessive packets sent to the VNF will cause VNF more congested, inhibiting the increase of p^{*VNF}. Hence, p^{*VNF} of VPH is larger than that of NPH at the beginning, while p^{*VNF} of VPH is smaller than that of NPH at the end and also p^{*VNF} increase in a logarithmic way.

The gap between *delay_NPH* and *delay_VPH* is fixed, no matter what p^N is used. This is because the packet arrival rates and other parameters for NPH and VPH are the same. Also, the equation pattern of average delay is the same for both cases in Eq. (1). The gap between *delay_NPH* and *delay_VPH* (shown in Fig. 6) is caused by changing the priority, but it is independent of p^N.

5.6 Impact of c^{VNF}

Figure 7 shows the impact of VNF service rate, c^{VNF}, on the average packet delay and p^{*VNF} for both cases. We can observe that with the increase of c^{VNF}, the packet delay decreases. When c^{VNF} increases, the average delay of packets requiring VNF definitely decreases, reducing the average delay. On the other hand, the increase of c^{VNF} will generate the increase of p^{*VNF} because with the increase of c^{VNF}, sending more packets to VNF becomes more beneficial.

As c^{VNF} increases, the gap between p^{*VNF} and the delay of VPH and NPH becomes larger. This is because in VPH, we have given higher priority to the packets that require VNF. With the increase of c^{VNF}, load decreases in the VNF queue more rapidly. Therefore, at the end, the optimal probability of going to VNF increases more rapidly for VPH. This phenomenon also decreases the average packet delay of VPH. Hence, the gap between p^{*VNF} and the delay of VPH and NPH increases.

5.7 Impact of DSV

Figure 8 shows that average packet delay increases as D^{SV} increases for both cases because the propagation delay to VNF increases.

However, average packet delay at VPH is still lower than that of NPH for the same reason described before. Also with the increase of D^{SV}, p^{*VNF} decreases because the benefits of offloading drops.

With the increase of D^{SV}, average packet delay increases fast in the beginning and becomes stable in the end. This is because propagation delay is only related to the packets that require network function at VNF (shown in Eq. (3)). As D^{SV} increases, benefits of offloading drops and after a particular point (30 μs), p^{*VNF} becomes 0. That means no packets are offloaded to VNF and therefore the effect of the D^{SV} diminishes. For this reason, the delay curve becomes stable at the end.

At the beginning with the increase of D^{SV}, the gap between *delay_NPH* and *delay_VPH* increases. This is because at the beginning VPH performs better than NPH for the same reason described before, thereby increasing the gap. However, at the end the gap is fixed, no matter what D^{SV} is used. This is because at the end the effect of the D^{SV} diminishes and the packet delay of VPH is lower than the packet delay of NPH because of the differences in prioritization value (shown in Eqs. (5), (6) and (9)).

6 Conclusion

In this paper, we presented a prioritization mechanism in the P4 + NFV architecture and found optimal probability of going to VNF from a P4 switch. Our study shows that NFV and P4 can be well-integrated to co-work together for minimizing average packet delay. We have calculated the average packet delay using an M/M/1 queuing model with non-preemptive priority analysis and justify our analysis with extensive simulations. Our experiment results displayed a significant performance gain of VPH compared to NPH. The optimal probability of going to VNF initially increases with the increase of λ, p^N, c^{VNF}. However, it falls after a particular value due to the communication-computation tradeoff.

Our evaluation revealed that under different packet arrival rates, optimal offloading from P4 switch to NFV can reduce the average packet delay from 13.74 to 40.73%, when packets requiring VNF are given a higher priority than newly coming packets. Furthermore, we observed that the improvement becomes more prominent as the packet arrival rate, the probability of requiring network functions, and VNF service rate increases. Hence, it can be concluded that VPH is better than NPH. We can say that the advantage of a prioritization mechanism in case of offloading from a P4 switch to NFV is significant. It helps to reduce the average packet delay of the system. By varying different parameters, sensitivity analysis of our experiment is also presented which may guide an operator to install a prioritization mechanism in the P4 + NFV architecture.

Author contributions Farhin Faiza Neha, Yuan-Cheng, Md. Shohrab Hossain and Ying-Dar Lin wrote the main manuscript text and Farhin Faiza Neha prepared figures and results. All authors reviewed the manuscript.

Funding No funding is applicable for this research.

Declarations

Competing interests The authors declare no competing interests.

References

1. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., & Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. In *23rd international teletraffic congress (ITC)*.
2. Papavassiliou. (2020). Software defined networking (SDN) and network function virtualization (NFV). *Future Internet, 12*(1), 7. https://doi.org/10.3390/fi12010007
3. Jawdhari, H. A., & Abdullah, A. A. (2021). The application of network functions virtualization on different networks, and its new applications in blockchain: A survey. *Webology, 18*(Special Issue 04), 1007–1044. https://doi.org/10.14704/web/v18si04/web18179
4. Goswami, B., Kulkarni, M., & Paulose, J. (2023). A survey on P4 challenges in software defined networks: P4 programming. *IEEE Access, 11*, 54373–54387. https://doi.org/10.1109/access.2023.3275756
5. Neha, F. F., Lai, Y. C., Hossain, M. S., & Lin, Y. D. (2023). Offloading in P4 switch integrated with multiple virtual network function servers. *Journal of Communications Software and Systems, 19*(4), 278–288. https://doi.org/10.24138/jcomss-2023-0125
6. He, M. (2018). P4NFV: An NFV Architecture with flexible data plane reconfiguration. In *14th International conference on network and service management (CNSM)* (pp. 90–98). IEEE.
7. Makara, L. A., Lai, Y. C., Lin, Y. D., Seah, W., & Pekar, A. (n.d.). Offloading from P4 Switches to Nfv in Programmable Data Planes. Available at SSRN 4090265.

8. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). Van wijngaarden-dekker-brent method. In *Numerical Recipes in FORTRAN: The art of scientific computing* (pp. 352–355). Cambridge University Press.
9. Mahmood, K., Chilwan, A., Østerbø, O. N., & Jarschel, M. (2014). *On the modeling of OpenFlow-based SDNs:The single node case*. https://doi.org/10.48550/ARXIV.1411.4733.
10. Mahmood, K., Chilwan, A., Østerbø, O., & Jarschel, M. (2015). Modelling of OpenFlow—Based software-defined networks: The multiple node case. *IET Networks, 4*(5), 278–284. https://doi.org/10.1049/iet-net.2014.0091
11. Nweke, L. O., & Wolthusen, S. D. (2020). Modelling adversarial flow in software-defined industrial control networks using a queueing network model. In *2020 IEEE Conference on Communications and Network Security (CNS)*.
12. Xiong, B., Yang, K., Zhao, J., Li, W., & Li, K. (2016). Performance evaluation of OpenFlow—Based software-defined networks based on queueing model. *Computer Networks, 102*, 172–185. https://doi.org/10.1016/j.comnet.2016.03.005
13. Zhao, J., Hu, Z., Xiong, B., Yang, L., & Li, K. (2020). Modeling and optimization of packet forwarding performance in software-defined WAN. *Future Generations Computer Systems: FGCS, 106*, 412–425. https://doi.org/10.1016/j.future.2019.12.010
14. Goto, Y., Ng, B., Seah, W. K. G., & Takahashi, Y. (2019). Queueing analysis of software defined network with realistic OpenFlow—Based switch model. *Computer Networks, 164*(106892), 106892. https://doi.org/10.1016/j.comnet.2019.106892
15. Singh, D., Ng, B., Lai, Y.-C., Lin, Y.-D., & Seah, W. K. G. (2018). Modelling software-defined networking: Software and hardware switches. *Journal of Network and Computer Applications, 122*, 24–36. https://doi.org/10.1016/j.jnca.2018.08.005
16. Singh, D., Ng, B., Lai, Y.-C., Lin, Y.-D., & Seah, W. K. G. (2020). Full encapsulation or internal buffering in OpenFlow based hardware switches? *Computer Networks, 167*(107033), 107033. https://doi.org/10.1016/j.comnet.2019.107033
17. Miao, W., Min, G., Wu, Y., & Wang, H. (2015). Performance modelling of preemption-based packet scheduling for data plane in software defined networks. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*.
18. Miao, W., Min, G., Wu, Y., Wang, H., & Hu, J. (2016). Performance modelling and analysis of software-Defined Networking under Bursty multimedia traffic. *ACM Transactions on Multimedia Computing Communications and Applications, 12*(5s), 1–19. https://doi.org/10.1145/2983637
19. Lin, P.-C., Lin, Y.-D., Wu, C.-Y., Lai, Y.-C., & Kao, Y.-C. (2016). Balanced service chaining with traffic steering in software defined networks with network function virtualization. *IEEE Computer, 49*(11), 68–76.
20. Fahmin, A., Lai, Y.-C., Hossain, M. S., & Lin, Y.-D. (2018). Performance modeling and comparison of NFV integrated with SDN: Under or aside? *Journal of Network and Computer Applications, 113*, 119–129. https://doi.org/10.1016/j.jnca.2018.04.003
21. Billingsley, J., Miao, W., Li, K., Min, G., & Georgalas, N. (2020). Performance analysis of SDN and NFV enabled mobile cloud computing. In *GLOBECOM 2020—2020 IEEE Global Communications Conference*.
22. Bi, C., Zheng, J., & Hu, Z. (2017). Hyper: A hybrid highperformance framework for network function virtualization. *IEEE Journal on Selected Areas in Communications, 35*(11), 2490–2500.
23. Paolucci, F., Cugini, F., Castoldi, P., & Osinski, T. (2021). Enhancing 5G SDN/NFV Edge with P4 data plane programmability. *IEEE Network, 35*(3), 154–160. https://doi.org/10.1109/mnet.021.1900599
24. Ji, S. (2020). DE4NF: High performance Nfv framework with P4-based event system (Doctoral dissertation).
25. Osinski, T., Tarasiuk, H., Rajewski, L., & Kowalczyk, E. (2019). DPPx: A P4-based Data Plane Programmability and Exposure framework to enhance NFV services. In *2019 IEEE conference on network softwarization (NetSoft)*.
26. Mohammad Khan, A., Panda, S., Kulkarni, S. G., Ramakrishnan, K. K., & Bhuyan, L. N. (2019). P4NFV: P4 enabled NFV systems with SmartNICs. In *2019 IEEE conference on network function virtualization and software defined networks (NFV-SDN)* (pp. 1–7). IEEE.
27. Zhang, T., Linguaglossa, L., Gallo, M., Giaccone, P., Iannone, L., & Roberts, J. (2019). Comparing the performance of state-of-the-art software switches for NFV. In *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*.
28. Palmer, G. I., Knight, V. A., Harper, P. R., & Hawa, A. L. (2019). Ciw: An open-source discrete event simulation library. *Journal of Simulation: JOS, 13*(1), 68–82. https://doi.org/10.1080/17477778.2018.1473909
29. Palmer, G. I., & Tian, Y. (2021). Implementing hybrid simulations that integrate DES+ SD in Python. *Journal of Simulation, 17*(3), 240–256.
30. Harkous, H., Jarschel, M., He, M., Pries, R., & Kellerer, W. (2021). P8: P4 with predictable packet processing performance. *IEEE Transactions on Network and Service Management, 18*(3), 2846–2859. https://doi.org/10.1109/tnsm.2020.3030102
31. Wang, S.-Y., Li, J.-Y., & Lin, Y.-B. (2020). Aggregating and disaggregating packets with various sizes of payload in P4 switches at 100 Gbps line rate. *Journal of Network and Computer Applications, 165*(102676), 102676. https://doi.org/10.1016/j.jnca.2020.102676

Farhin Faiza Neha Farhin Faiza Neha has completed M.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET). She has completed B.Sc. in Computer Science and Engineering from Chittagong University of Engineering & Technology (CUET), Farhin actively contributes to the 'Establishing Digital Connectivity (EDC)' project as an Assistant Network Engineer at the Department of ICT, Govt. of Bangladesh. Her research interests include wireless networks communication, network performance evaluation, cyber security, machine learning, artificial intelligence, internet of things, and connected and autonomous vehicular systems.

Yuan-Cheng Lai received the Ph.D. degree in Computer Science from National Chiao Tung University in 1997. In August 2001, he joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology where he had been a professor since February 2008. His research interests include wireless networks, network performance evaluation, network security, and Internet applications.

Md. Shohrab Hossain received his B.Sc. and M.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in the year 2003 and 2007, respectively. He obtained his Ph.D. degree from the School of Computer Science at the University of Oklahoma, Norman, OK, USA in December, 2012. During his Ph.D., he worked under NASA funded projects related to survivability, scalability and security of space networks. He is currently serving as a Professor in the Department of Computer Science and Engineering at Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. His research interests include Cyber security, Mobile malware detections, Software defined networking (SDN), security of mobile and ad hoc networks, and Internet of Things. He has published more than 98 technical research papers in leading journals and conferences including Journal of Computers & Security, Ad Hoc Networks, IEEE Access, Journal of Network and Computer Applications, Journal of Telecommunication Systems, Wireless Personal Communication, PLOS ONE, IEEE GLOBECOM, IEEE ICC, IEEE MILCOM, IEEE WCNC, IEEE HPCC, etc. He has been serving as the TPC member of IEEE GLOBECOM, IEEE ICC, IEEE VTC, Wireless Personal Communication, Journal of Network and Computer Applications, IEEE Wireless Communications.

Ying-Dar Lin (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), in 1993. Since 2002, he has been the Founder and the Director of the Network Benchmarking Laboratory. He is currently a Chair Professor of computer science at the National Yang Ming Chiao Tung University (NYCU), Taiwan. He published a textbook, Computer Networks: An Open Source Approach. His research interests include network security, wireless communications, and network softwarization. He has served or is serving on the editorial boards for several IEEE journals and magazines, and was the Editor-in-Chief of the IEEE Communications Surveys and Tutorials, during 2017–2020.