Offloading in P4 Switch Integrated with Multiple Virtual Network Function Servers

Farhin Faiza Neha, Yuan-Cheng Lai, Md. Shohrab Hossain, and Ying-Dar Lin

Original scientific article

***Abstract*—Software Defined Networking (SDN) and Network Function Virtualization (NFV) are two transformative technologies that offer distinct benefits. SDN virtualizes the control plane by separating it from the data plane, while NFV virtualizes the data plane by moving network functions from hardware and implementing them in software. Therefore, combining SDN and NFV can fully exploit the benefits of both technologies. As Programming Protocol-independent Packet Processors (P4) become popular due to its flexibility, traditional SDN switches are being replaced by P4 switches. In the P4+NFV architecture, network functions can be provided in both P4 switches (PNF) and NFV servers (VNF). However, to minimize packet delay, the offloading problem between P4 switches and NFV needs to be addressed. The novelty of our paper lies in investigating the offloading problem and evaluating the impact of employing multiple VNFs with varying computing capacities within the P4+NFV architecture. We also use M/M/1 queuing theory to derive the average packet delay and propose an optimization solution based on gradient descent to find out the optimal offloading probabilities of various VNF servers. Results show that optimal offloading from P4 switch to multiple VNFs can reduce the average packet delay from 4.76% to 40.02%.**

***Index Terms*—software defined networking, network function virtualization, virtual network functions, optimal probability.**

I. Introduction

Traditional network architectures often encounter challenges when updating hardware devices like routers and switches due to their inflexible nature. Nevertheless, SDN provides a promising solution to this issue. SDN fundamentally separates the control plane from the data plane, entrusting a controller with decision-making regarding the optimal data path from source to destination within the control plane. In contrast, the switches in the data plane simply forward packets based on the controller's decisions [1]. SDN virtualizes the network's control plane, presenting a more flexible and dynamic approach to network management.

On the contrary, network functions (NFs) such as deep packet inspection and load balancing are traditionally provided through specialized hardware integrated with software. However, this hardware can often prove costly and cumbersome to update. In response to these challenges, a new technology known as Network Function Virtualization (NFV) has emerged [2]. NFV accomplishes this by virtualizing these functions moving them from dedicated hardware to software installed on readily available server [3]. This approach enables NFs to be implemented with increased flexibility and cost-effectiveness, fostering the creation of service function chaining [4]. NFV virtualizes the data plane and creates modular components that can be interconnected to support various NFs. In contrast, SDN virtualizes the control plane. Therefore, integrating SDN with NFV can provide an excellent architecture that combines the advantages of both technologies.

In recent years, there have been significant advancements in programmable switch chips, enabling the processing of packets at high speeds similar to fixed-function switches. This progress has given rise to Programming Protocol-Independent Packet Processors (P4), a technology that utilizes a domain-specific programming language in conjunction with an SDN controller to achieve protocol-independence, target-independence, and field configuration [5]. P4 empowers programmers to specify recognized input packet headers, define map-action tables and processing algorithms. Furthermore, P4's programmability equips it to handle certain network functions.

Due to the advantages offered by P4 switches over traditional switches, they are experiencing growing demand. P4 switches may soon become the preferred choice in networking. Consequently, the architecture that combines SDN with NFV may shift from traditional switches to P4 switches, resulting in the integration of P4 switches and NFV (referred to as P4+NFV). In the P4+NFV architecture, NFs can be provided in both P4 switches and NFV. Therefore, an important consideration involves determining the appropriate allocation of packets, deciding which ones should be forwarded to NFV for Virtualized Network Functions (VNFs), and which should remain within P4 switches for Physical Network Functions (PNFs). Addressing this offloading problem is crucial for creating more flexible, scalable, resilient, and cost-efficient network architectures while reducing average packet delay.

In prior research study, He et al. [6] proposed a hybrid architecture that combines P4 switches with NFV to achieve greater flexibility and speed, meeting the demands of modern network bandwidth requirements. Makara et al. [7] analyzed the impact of offloading probability (from P4 switches to NFV) on

Manuscript received October 12, 2023; revised November 20, 2023. Date of publication December 1, 2023. Date of current version December 1, 2023.

F. F. Neha and Md. S. Hossain are with the Bangladesh University of Engineering and Technology, Bangladesh (e-mails: farhinfaiza@gmail.com, mshohrabhossain@cse.buet.ac.bd).

Y. - C. Lai is with the National Taiwan University of Science and Technology, Taiwan (laiyc@cs.ntust.edu.tw).

Y. - D. Lin is with the National Yang Ming Chiao Tung University, Hsinchu, Taiwan (e-mail: ydlin@cs.nctu.edu.tw).

Digital Object Identifier (DOI): 10.24138/jcomss-2023-0125

various performance metrics, using Brent's method [8]. While some recent studies [6-7] have explored the combination of P4 switches with NFV, none has delved into investigating the performance gains of this hybrid architecture when employing multiple VNFs with varying computing capacities for different VNF queues. This novel aspect forms the foundation of our work. We employ multiple VNFs to compare their optimal offloading probabilities, enhancing data processing efficiency and resulting in a significant reduction in average packet delay. Furthermore, we evaluate the performance metric of packets requiring multiple VNFs using an M/M/1 queuing model in the P4+NFV architecture. It is important to note that the conventional approach typically employs the P4 switch as the default data plane, offering the option to offload traffic to VNF when congestion occurs. 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.

This paper makes several contributions, including: (i) developing an analytical model using an M/M/1 queuing model to analyze the P4+NFV architecture with multiple VNF servers, (ii) proposing an algorithm to determine the optimal offloading probabilities from P4 switch to multiple VNFs, (iii) investigating different VNF computing capacities for different VNF servers to determine their optimal offloading probabilities, and (iv) evaluating the offloading from P4 to multiple VNFs in terms of various performance metrics under different parameter settings.

The rest of the paper is organized as follows. Section II provides an overview of previous related works. Section III presents the system model of the P4+NFV architecture utilizing multiple VNFs. Section IV derives the average packet delay analytically and describes the algorithm for finding the optimal offloading probabilities. Section V presents analytical and simulation results to demonstrate the performance of using multiple VNFs with varying VNF computing capacities. Finally, Section VI concludes the paper.

II. Related Works

There have been a few studies on integrating P4-based programmable switches with NFV. Although some previous studies have analyzed the performance of SDN/NFV, there is limited investigation into the performance benefits of the P4+NFV architecture using an M/M/1 queuing model with multiple VNFs employing different service rates for distinct VNF queues. Table I summarizes the key findings from previous studies across four categories: *SDN (traditional switch), multiple VNFs, SDN (traditional switch)+NFV,* and *SDN (P4 switch)+NFV.*

1) SDN (traditional switch): Raychev et al. [9] developed an M/M/1 queuing model for both SDN switches and controllers to manage data traffic. Sarkar et al. [10] proposed an OpenFlow-based SDN switch using M/M/1 queuing theory and exponential models. Nweke et al. [11] employed an M/M/1 queuing model to analyze the consequences of adversarial flow in an SDN infrastructure. Goto et al. [12] introduced a queuing model for OpenFlow-based SDN switches, focusing on error minimization and validation in a test environment. Singh et al. conducted studies on the trade-offs between software and hardware switches [13] and the encapsulation versus internal buffer usage [14] in UDP using continuous-time Markov chains. However, none of these previous studies [9-14] have specifically addressed the performance of the P4+NFV hybrid architecture concerning multiple VNFs with varying service rates for different VNF queues. Furthermore, this paper argues that software and hardware are not competitors but can be effectively integrated.

2) Multiple VNFs based techniques: Nikolai et al. [15] analyzed the performance improvement of x86 hosts with multiple VNFs, highlighting significant throughput differences between single and multiple VNF systems. Quang et al. [16] formulated an optimization problem using integer linear programming (ILP) and introduced a heuristic algorithm for the allocation of multiple virtual network function-forwarding graphs. Yamada et al. [17] introduced Service Function Chain (SFC) to address challenges associated with the utilization of multiple VNFs. Rossem et al. [18] developed an efficient method for VNF chain deployment to interconnect multiple VNFs, reducing iterations and streamlining the time-consuming VNF chain validation process. However, none of these studies [15-18] have examined the use of multiple VNFs with different computing capacities in a P4+NFV-based hybrid architecture or compared their optimal offloading probabilities.

3) Combination of SDN (traditional switch) and NFV based techniques: Ramya et al. [19] developed a traffic management model to predict the optimal number of controllers to deploy within SDN+NFV architectures. Fahmin et al. [20] introduced a hybrid architecture that combines SDN and NFV, investigating the optimal placement of NFV in relation to the controller. They employed M/M/1 queuing theory to calculate average packet delay. Billingsley et al. [21] proposed a model for analyzing the performance of Mobile Cloud Computing within SDN+NFV architectures using M/M/1 queuing theory. Surantha et al. [22] improved the performance and functionality of NFV devices by integrating them with SDN, replacing the standard virtual switch with a data plane development kit, and implementing single-root I/O virtualization technology.

4) Combination of SDN (P4 switch) and NFV based techniques: He et al. [6] were the first to propose a P4 switch and NFV-based hybrid architecture that offers increased flexibility and faster speed. Therefore, this hybrid architecture is better suited to current network bandwidth requirements.

Paolucci et al. [23] introduced a method for integrating P4 Data Plane Programmability (DPP) into SDN/NFV. This method enhances flexibility in a range of applications, including 5G networks, IoT, cyber security, and traffic engineering. Ji et al. [24] investigated a high-performance event system combining NFV with a P4 switch capable of supporting multiple function chains at line rate and reducing packet delay. Osiński et al. [25] utilized the BMv2 software switch and exposure framework (DPPx) to propose a P4-based Data Plane Programmability model improving the flexibility of NFV

TABLE I
SUMMARY OF RELATED WORKS

Category	*References*	*NF queues used?*		*Characteristics*
		# of VNFs	*PNF*	
SDN (traditional switch)	[9]	0	No	Developed a model for both SDN switch and controller using M/M/1 theory
	[10]	0	No	Developed an OpenFlow-based model for both SDN switch using M/M/1 theory
	[11]	0	No	Adversarial flow using M/M/1 theory
	[12]	0	No	Priority based solution with Markov Chain 2D MC (HPQ,LPQ)
	[13]	0	No	Prioritization in 2D MC (HPQ, LPQ) and software vs. hardware switches
	[14]	0	No	Prioritization in 4D MC (internal buffer, HPQ, LPQ, hardware) and encapsulation vs. internal buffer
Multiple VNFs	[15]	N	No	Performance gain of an x86 host running multiple VNFs
	[16]	N	No	A heuristic algorithm for allocating multiple VNFs
	[17]	N	No	Introduced SFC which is comprised of multiple VNFs
	[18]	N	No	VNF chain deployment for connecting multiple VNFs
SDN (traditional switch)+NFV	[19]	1	No	Traffic management model to predict the optimal number of controllers
	[20]	1	No	Combination of SDN and NFV using M/M/1
	[21]	1	No	Mobile Cloud Computing using M/M/1 theory
	[22]	1	No	Replaced the standard virtual switch with a data plane development kit
SDN (P4 switch)+NFV	[6]	1	Yes	Fundamental modeling of P4+NFV architecture
	[23]	1	No	5G SDN/NFV Edge with P4 switch
	[24]	1	No	NFV framework with event system based on P4 switches
	[25]	1	Yes	P4-based Data Plane Programmability and Exposure framework (DPPx)
	[26]	1	Yes	MILP based method for boosting capacity in SmartNICs
	[27]	1	Yes	Seven state-of-the-art software switches for offloading NFV traffic between NICs and VNFs
	[7]	1	Yes	Impact of offloading from P4 switches to NFV using Brent's method
	Our proposed model	N	Yes	Combination of P4 and NFV using multiple VNFs and M/M/1 queuing theory

services. Additionally, the P4+NFV-based hybrid architecture has shown potential in enhancing the performance of network interface cards (NICs). Mohammad et al. [26] introduced a Mixed Integer Linear Programming (MILP) based optimization method using P4+NFV architecture for SmartNICs, effectively reducing packet delay and increasing flexibility. Zhang et al. [27] conducted a study on the performance gain of NFV in offloading traffic between physical NICs and VNFs.

However, no P4+NFV architecture has yet been developed to determine the optimal offloading probability to reduce packet delay when using multiple VNFs with different computing capacities for different VNF queues. Makara et al. [7] studied the impact of offloading probability on various performance metrics using Brent's method [8]. They used a controller to decide the route and required operation of a specific packet. In contrast, our proposed approach analyzes the impact of offloading probabilities in the context of using multiple VNFs, entirely eliminating the need for a controller. Dependency on a central controller might introduce a single point of failure. Removing the controller could enhance system robustness by distributing decision-making processes. Without a controller, the architecture may be simpler and more streamlined.

Our work is most relevant to the category of *SDN (P4 switch)+NFV*. Previous studies have not investigated performance benefits of this architecture when using multiple VNFs with different computing capacities. Existing studies have focused on *SDN (traditional switch), multiple VNFs, SDN+NFV, and SDN (P4 switch)+NFV* separately. Some studies have analyzed aspects such as load balancing, VNF placement, and performance improvements. However, none of these studies have specifically investigated the performance of the P4+NFV architecture when using multiple VNFs with different computing capacities for distinct VNF queues. This research aims to fill this gap and contribute to a better understanding of the optimal offloading probabilities and performance gains in the P4+NFV architecture with multiple VNFs.

III. SYSTEM MODEL

In our system model, when traffic reaches a switch and requires a NF, it has two possible paths. It can either be processed within the P4 switch, referred to as a PNF (Physical Network Function), or it can be routed to the NFV located in the data center. If the packets are sent to the VNF, they will return to the switch before being directed to their final destination.

A. System Model

Fig. 1 shows a network model of a programmable P4-based switch with multiple VNFs. To calculate the average packet delay, we have used the M/M/1 queuing model. The network model consists of several queues, including a switch processing (SP) queue, a PNF queue, a switch communication (SC) queue, and multiple VNFs queues:

- *A switch processing (SP) queue:* The SP queue is responsible for processing all packets, including new ones (shown in black) and those that have visited the VNF and re-entered the switch (shown in blue).
- *A PNF queue:* The PNF queue processes packets that require a network function inside the switch (shown in red).
- *A switch communication (SC) queues:* The SC queue forwards packets to their next hop, including newly arriving packets (shown in black) and those that require processing by VNFs (shown in blue).

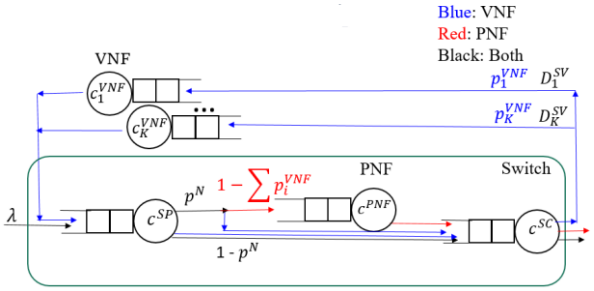

Fig. 1. Queuing model of a programmable switch using multiple VNF servers.

- *Multiple VNFs queues*: Lastly, packets that require VNF functions (shown in blue) are queued in the multiple VNFs queues, which are then processed and sent back to the switch.

Based on the network model shown in Fig. 1, when new packets arrive at the switch (indicated by the incoming arrow on the left), they are first processed by the P4 switch. The switch processing module then checks whether the packet requires a NF or not. If it does, the module decides whether to process the packet in the P4 switch itself (PNF) or offload it to one of the multiple VNFs based on their respective offloading probabilities. Following this decision, the packets receive the necessary VNF services. After that, they re-enter the switch and are directed to their destination by the switch communication module.

The model makes some assumptions, including (1) Packet arrivals at the switch follow a Poisson process, (2) The size of each queue is infinite, (3) NF is required only once, (4) There are multiple VNF queues, and there is no separation of new packets and those that have already undergone NF in the input queue.

B. Notations Used in the Analysis

Table II 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 at the i^{th} VNF queues are denoted as c^{PNF} and c_i^{VNF}, respectively. The packets arrive at the P4 switch at a rate λ. The number of VNF queues used in the analysis is represented by k. 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 i^{th} VNF queue is represented by p_i^{VNF}. Our goal is to determine the optimal offloading probabilities to the VNF queues, denoted as p_i^{*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 section IV). 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 i^{th} VNF queue. Thus, we use the notations t^{SP}, t^{SC}, t^{PNF}, and t_i^{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_i^{SV} represents the fixed propagation delay at the i^{th} VNF queue.

TABLE II
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 the PNF queue
	c_i^{VNF}	Service rate or computing capacity at the i^{th} VNF queue
Arrival rate	λ	Packet arrival rate
Number of VNFs	k	Number of VNF queues
Probability	p^N	Probability of requiring network function
	p_i^{VNF}	Probability of going to the i^{th} VNF queue
Packet delay for three types of packets	d^{PNF}	Packet delay for packets that require network function at PNF
	d^{VNF}	Packet delay for packets that require network function at VNF
	d^{ONF}	Packet delay for packets that do not require any network function
Average packet delay at queues	t^{SP}	Average packet delay at SP queue
	t^{SC}	Average packet delay at SC queue
	t^{PNF}	Average packet delay at PNF queue
	t_i^{VNF}	Average packet delay at i^{th} VNF queue
Delay	D	Total average packet delay
	D_i^{SV}	Fixed propagation delay at i^{th} VNF queue

C. 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};
- Switch computing capacity: c^{PNF};
- VNF computing capacity at the i^{th} queue: c_i^{VNF};
- Packet arrival rate: λ;
- Probability of requiring network functions: p^N;
- Fixed propagation delay at the i^{th} VNF queue: D_i^{SV};
- The number of VNFs: k

Output:

- Optimal probability of going to i^{th} VNF queue: p_i^{*VNF}

Objective:

- Minimize the average packet delay D.

That is, the problem is to find the optimal p_i^{VNF}, denoted as p_i^{*VNF} using multiple VNFs.

Constraint:

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

IV. SOLUTION

There are two main goals in this work: (1) to calculate the average packet delay in a P4-based switch with multiple VNFs using an M/M/1 queuing model, and (2) to propose an optimization algorithm that finds the optimal offloading probabilities for packets to go to multiple VNFs based on the

derived formulas for average packet delay.

A. Average Packet Delay

We now calculate the total average packet delay of the proposed queuing network. To do so, we will first calculate the arrival and service rates at each queue shown in Fig. 1, and then use the M/M/1 theory to find the total average packet delay. We have listed the notations used in our analytical model in Table II.

For mathematical analysis, we have considered three types of packets:

1) Packets that require network function at PNF,
2) Packets that require network function at VNF,
3) Packets that do not require any network function.

Then, we calculate packet delay for all of these three types of packets.

1) Packets that require network function at PNF: Packets destined for PNF pass through the SP queue, the PNF queue, and finally the SC queue."

a) Delay at the SP queue: Initially, packets enter the switch at an arrival rate λ through the switch's processing queue SP. Additionally, packets that have undergone processing by multiple VNFs re-enter the switch via the SP queue at an arrival rate ($\lambda p^N \sum_{i=1}^{K} p_i^{VNF}$). Here, the probability of requiring network function is p^N and total probability of going to all the VNF queues is $\sum_{i=1}^{K} p_i^{VNF}$. Hence, arrival rate at SP queue denoted as λ^{SP} can be calculated as

$$\lambda^{SP} = \lambda + \lambda p^N \sum_{i=1}^{K} p_i^{VNF}. \quad (1)$$

Using M/M/1 queuing theory, the average packet delay at the switch's processing (SP) queue, denoted as t^{SP}, can be calculated as

$$t^{SP} = \frac{1}{c^{SP} - \lambda^{SP}}, \quad (2)$$

where, c^{SP} is the service rate at the SP queue.

b) Delay at the PNF queue: Some packets that require network function traverses the PNF queue. Therefore, the arrival rate at the PNF queue, denoted as λ^{PNF}, is expressed as

$$\lambda^{PNF} = \lambda p^N (1 - \sum_{i=1}^{K} p_i^{VNF}). \quad (3)$$

Using M/M/1 queuing theory, the average packet delay at the switch's PNF queue can be calculated as

$$t^{PNF} = \frac{1}{c^{PNF} - \lambda^{PNF}}. \quad (4)$$

where, c^{PNF} is the service rate at the PNF queue.

c) Delay at the SC queue: Packets leave the system through the switch's communication queue SC. Packets which does not require any network function at VNF, leave the switch using the SC queue with an arrival rate λ and packets which require VNF exit through the SC queue with an arrival rate ($\lambda p^N \sum_{i=1}^{K} p_i^{VNF}$). Hence, the arrival rate at SC queue, denoted as λ^{SC}, is expressed as

$$\lambda^{SC} = \lambda + \lambda p^N \sum_{i=1}^{K} p_i^{VNF}. \quad (5)$$

If service rate at the SC queue is c^{SC}, then using M/M/1 queuing theory the average packet delay at the switch's communication (SC) queue can be calculated as

$$t^{SC} = \frac{1}{c^{SC} - \lambda^{SC}}. \quad (6)$$

So, packet delay for packets that require network function at PNF is

$$d^{PNF} = (t^{SP} + t^{PNF} + t^{SC}). \quad (7)$$

2) Packets that require network function at VNF: Packets requiring VNFs traverse both the SP and SC queues twice and the VNF queues once. We have already provided calculations for the average packet delay at the SP and SC queues in (2) and (6). Consequently, our focus here is on deriving the equation for calculating the average packet delay specifically for the VNF queues.

a) Delay at the VNF queues: Some packets that require network functions at VNFs first pass through the SP and SC queues before traversing the VNF queues. In this scenario, we make use of multiple VNFs. Hence, arrival rate for the i^{th} VNF queue, denoted as λ_i^{VNF}, is expressed as

$$\lambda_i^{VNF} = \lambda p^N p_i^{VNF}. \quad (8)$$

As depicted in Fig. 1, packets that pass through the VNF queues must return to the switch, incurring a $2D_i^{SV}$ propagation delay. If the service rate at the i^{th} VNF queue is c_i^{VNF}, then, accounting for the fixed propagation delay, the average packet delay at the i^{th} VNF queue is expressed as

$$t_i^{VNF} = \frac{1}{c_i^{VNF} - \lambda p^N p_i^{VNF}} + 2D_i^{SV}. \quad (9)$$

Considering the total probability of going to all VNF queues as $\sum_{i=1}^{K} p_i^{VNF}$, we can determine the delay for packets requiring network functions at VNFs as

$$d^{VNF} = \sum_{i=1}^{K} p_i^{VNF} (2t^{SP} + t_i^{VNF} + 2t^{SC}). \quad (10)$$

3) Packets that do not require any network function: Packets that do not require any network function pass through the SP and SC queues once. We have already provided calculations for the average packet delay at the SP and SC queues in (2) and (6), respectively. Consequently, the packet delay for packets not requiring any network function is determined as

$$d^{ONF} = (t^{SP} + t^{SC}). \quad (11)$$

Finally, from (7), (10) and (11), according to the ratios of packets we can obtain the average packet delay, as

$$D = p^N \left(1 - \sum_{i=1}^{K} p_i^{VNF}\right) d^{PNF} + p^N d^{VNF} + (1 - p^N) d^{ONF}. \quad (12)$$

Subsequently, our proposed algorithm explores the state space to determine the optimal probabilities (p_i^{*VNF}) for directing packets to the VNFs. This optimization aims to minimize the average packet delay as defined in (12).

B. Algorithm for Finding Optimal Offloading Probabilities

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

To begin, we initialize p_i^{VNF} with the value of p^{INI} (initial probability) divided by *k* (number of VNFs). Two parameters: *step* and *stepRF*, are used to control the searching range and gradually reduce the step size in each iteration, respectively. Four arrays: *minscope[], maxscope[], optimal[], and x[],* are declared. The PINOpt algorithm searches for optimal values of p_i^{VNF} by generating a search space [*minscope*, *maxscope*] in each step. The *min* and *max* values are appended to the *minscope[]* and *maxscope[]* arrays, respectively, as shown in lines 8 to 12. Moving on to lines 13 to 21, the algorithm finds the optimal p_i^{VNF} values that correspond to the minimum delay within the defined search space. Initially, the values of *minscope[]* are copied to the *optimal[]* array. The *minDelay* (minimum delay) is initially set to *Infinity*. Then, we assign the i^{th} value of the *minscope[]* array to the i^{th} value of the *x[]* array. The *while* loop continues until the i^{th} value of the *x[]* array exceeds the i^{th} value of the *minscope[]* array. Within the loop, the *DelayforProb(x[i])* function calculates the average packet delay based on the p_i^{VNF} values stored in the *x[]* array.

Equation (12) is used in the *DelayforProb(x[i])* function to calculate the total average packet delay. The necessary parameter values for the *Delay* (total average packet delay) calculation are listed in Table III. During the calculation, if the computed *Delay* value is less than the current *minDelay*, *minDelay* is updated accordingly. The optimal p_i^{VNF} values, which correspond to the minimum delay, are stored in the *optimal[]* array. As we approach the solution, the *x[]* array is updated using the step value. This *step* value gradually decreases, controlled by the *stepRF* parameter. The required accuracy for finding the optimal p_i^{VNF} values is controlled by an input parameter ε (precision).

Algorithm 1: PINOpt for finding optimal $\boldsymbol{p_i^{VNF}}$ values

Input: $\boldsymbol{\varepsilon}$, p^{INI} , *stepRF, k*
Output: $\boldsymbol{p_1^{VNF}}$, $\boldsymbol{p_2^{VNF}}$, … , $\boldsymbol{p_k^{VNF}}$

```
1  step = p^INI / stepRF
2  minscope=[]
3  maxscope=[]
4  optimal=[]
5  x = []
6  while  i ≤ k : do
7      p_i^VNF = p^INI/k
8      while ( step ≥ ε ) : do
9          min = max [0, p_i^VNF - (step×stepRF)]
10         minscope.append (min)
11         max = min [1, p_i^VNF + (step×stepRF)]
12         maxscope.append (max)
13         optimal[i]= minscope[i]
14         minDelay = ∞
15         x [i] = minscope[i]
16         for i in range (len(x)):
17             while x [i] ≤ maxscope[i]: do
18                 Delay = DelayforProb(x[i])
                       /* Eqn. (12) according to x[i]
19                 if (Delay < minDelay): then
20                     minDelay = Delay
21                     optimal[i]= x[i]
22                 x[i] + = step
23             p_i^VNF =optimal[i]
24         step = step / stepRF
25     return p_i^VNF
26 i=i+1
27 end
```

V. Analytical and Simulation Results

A. Designing a Custom Simulator

We have designed a custom simulator using the Ciw event simulation library [28-29] to validate our analytical model. Fig. 2 shows the flowchart of the simulation process. This simulator generates packets following the Poisson distribution and routes them through different queues based on specific probabilities. Poisson distribution is commonly used to model packet arrivals in network traffic. This distribution is often applied to events that occur randomly and independently over time. In our simulator, we have implemented a packet class and routing function to handle the packet routing process. We have assumed fixed packet sizes for reducing overhead associated with variable-sized packets. In variable-sized packet systems, the headers and metadata required for each packet can vary, potentially leading to increased overhead. Fixed-sized packets ensure a consistent overhead for each packet, promoting efficiency. The routing function assigns packets to specific queues based on probabilities, and we use an event queue to ensure sequential processing of queuing events. To evaluate system performance, we log each packet's path through the queues, enabling us to calculate average packet delay. We plan to analyze this data to optimize the probabilities for sending packets to VNF queues from the P4 switch, with the goal of minimizing average packet delay. By integrating our analytical model with simulation results from the custom simulator, we can comprehensively understand the system's behavior and validate our proposed approach's effectiveness.

B. Parameter Settings

Table III presents the baseline parameters used in both the analysis and simulation. In a typical OpenFlow network, the probability of requiring a network function (p^N) is assumed to be 50%. The VNF service rate (c_1^{VNF}) is set to 95,000 packets/sec, representing a low VNF computing capacity [20]. The packet arrival rate (λ) is 125,000 packets/sec [30]. To examine the impact of varying VNF service rates on performance, we consider a high capacity of 950,000 packets/sec for c_2^{VNF} and a very high capacity of 9,500,000 packets/sec for c_3^{VNF}. The switch communication rate (c^{SC}) is fixed at 1 Gbps (625,000 packets/sec), while the switch processing rate (c^{SP}) is set at 12,500,000 packets/sec, as indicated by previous studies [30-31]. Each simulation is repeated 100 times, and the average values of the metrics are recorded.

Initially, we demonstrate the impact of p_i^{VNF} on the average packet delay to highlight its significance. Subsequently, we analyze the sensitivity of various parameters, including λ, p^N, c_i^{VNF}, and D_i^{SV}.

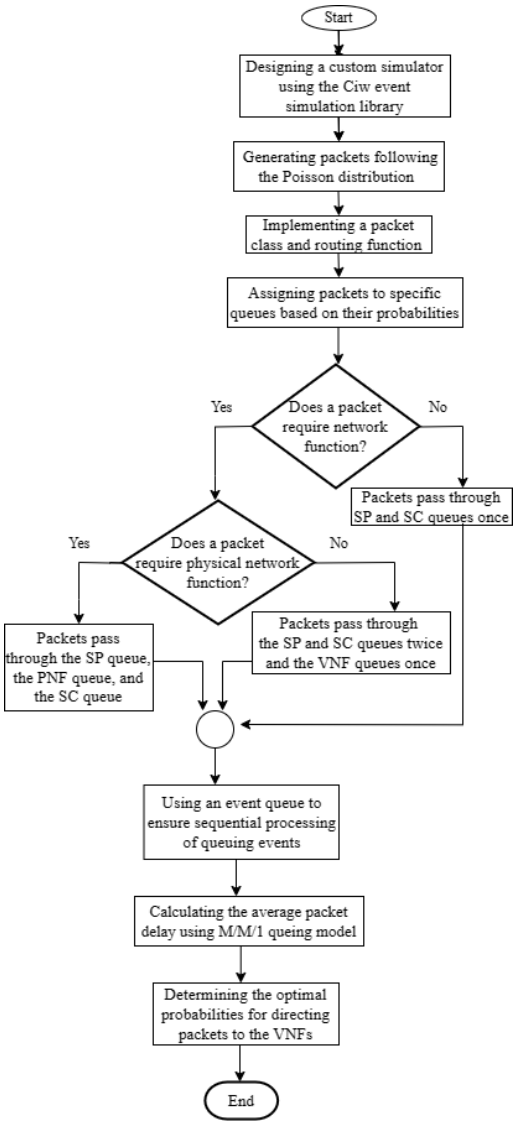

Fig. 2. Flowchart of the simulation process.

C. Optimal Values of p_i^{VNF}

Fig. 3 shows the impact of probabilities for routing packets to multiple VNFs (p_i^{VNF}) on average packet delay, considering k = 2. The plot depicts the relationship between p_1^{VNF}, p_2^{VNF}, and the average packet delay. Notably, our analytical results closely align with the simulation results, confirming the precision of our analytical model in replicating real-world scenarios. Optimal p_i^{VNF} values vary with the number of VNFs. Consequently, as k changes, p_i^{VNF} values also shift. We systematically varied p_1^{VNF} from 0 to 1 while applying the same range to p_2^{VNF}, with the constraint that $p_1^{VNF} + p_2^{VNF} \leq 1$. In this scenario, we assigned a low capacity of 95,000 packets/sec to c_1^{VNF} and a high capacity of 950,000 packets/sec to c_2^{VNF}. The average packet delay plot shows that when both p_1^{VNF} and p_2^{VNF} are 0 (indicating no offloading to VNFs), severe congestion occurs in the PNF queue, leading to higher packet delays. As the probabilities of using VNFs (p_1^{VNF} and p_2^{VNF}) increase, average packet delay decreases. This is because offloading packets to VNFs reduces the load in the PNF queue, easing congestion. The lowest average packet delay (7.69 µs for the analytical model and 7.91 µs for the simulation) occurs when p_1^{VNF} is 0.175810 and p_2^{VNF} is 0.289530.

TABLE III
BASELINE PARAMETERS FOR THE ANALYSIS AND SIMULATION

Symbol	*Value*
c^{SP}	12,500,000 pkts/sec
c^{SC}	625,000 pkts/sec
c^{PNF}	125,000 pkts/sec
λ	125,000 pkts/sec
k	3
c_1^{VNF}	95,000 packets/sec
c_2^{VNF}	950,000 packets/sec
c_3^{VNF}	9,500,000 packets/sec
p^N	0.5
D_i^{SV}	10 µs
ε	10^{-6}
p^N	0.5
stepRF	10

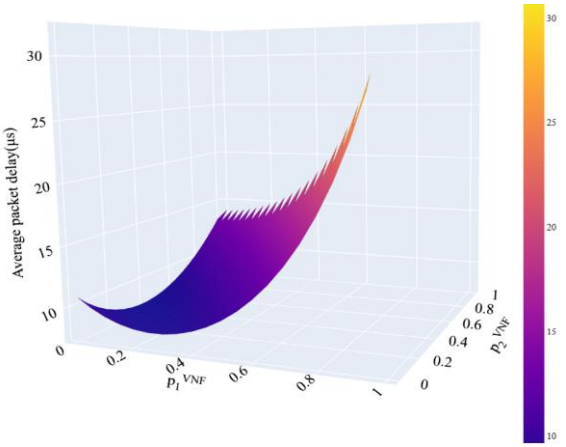

Fig. 3. Impact of p_i^{VNF} on the average packet delay.

Additionally, it is observed that p_2^{*VNF} is higher than p_1^{*VNF}. This discrepancy arises due to the different VNF service rates (c_i^{VNF}) for each VNF queue. A higher c_i^{VNF} value results in a reduced average delay for packets requiring VNF processing. Consequently, higher c_i^{VNF} values lead to higher p_i^{*VNF} values, as sending more packets to the VNF queues becomes more advantageous.

Beyond a certain threshold, as p_1^{VNF} and p_2^{VNF} increase, the average packet delay again tends to increase. This is due to the increased number of packets routed through VNFs, which introduces additional overhead and consequently leads to higher delays associated with processing packets through multiple VNFs.

In the following results, we investigate the sensitivity of various parameters for k = 3. We examine the impact of offloading packets from the P4 switch to multiple VNFs, considering aspects such as λ, p^N, c_i^{VNF}, and D_i^{SV}. Our study offers valuable insights and recommendations for service providers dealing with latency issues and tackling design and

service administration challenges when utilizing multiple VNFs with distinct service rates for each VNF queue.

To facilitate the comparison and analysis of the impact of different numbers of VNFs on the average packet delay, we denote the average packet delay for $k = 1$ as *delay_k(1)* and for $k = 3$ as *delay_k(3)*.

D. Impact of Packet Arrival Rate, λ

Fig. 4 demonstrates the relationship between the arrival rate (λ) and the optimal offloading probabilities (p_i^{*VNF}) as well as the average packet delay. The results obtained from both the analytical model and simulation align closely. In Fig. 5, we compare the packet delay when using a single VNF versus using multiple VNFs.

As shown in Fig. 4, an increase in λ leads to a significant increase in p_i^{*VNF}, causing more packets to be offloaded to all three VNFs based on their VNF service rates (c_i^{VNF}). This is because, as λ increases, the packet delay for packets requiring PNF increases at a faster rate compared to the packet delay for packets requiring VNFs (as shown in Fig. 6 and will be explained later). Consequently, it becomes more advantageous to offload a greater portion of packets to VNFs, resulting in higher offloading probabilities for all three VNFs. When network traffic is low, offloading packets to VNFs may not provide significant benefits since the PNF can handle incoming packets without congestion. However, as network traffic load increases, it becomes more advantageous to offload packets to VNFs to alleviate congestion at the PNF queue. In summary, the optimal offloading probabilities (p_i^{*VNF}) and average packet delay are influenced by the packet arrival rate (λ). With increasing λ, p_i^{*VNF} increases notably as more packets are offloaded to VNFs to prevent congestion at the PNF queue. However, there comes a point where the slope of the curve starts to decrease, indicating diminishing returns from further offloading. This occurs because excessively offloading packets to VNFs can lead to congestion at the VNF queues, reducing the benefits of offloading. Overall, it is more effective to offload packets to VNFs as the load at the PNF queue increases.

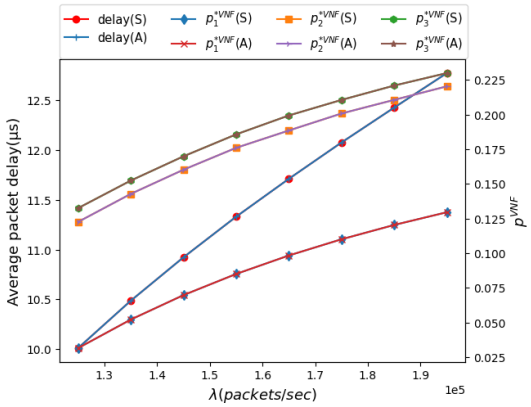

Fig. 4. Impact of arrival rate, λ on delay and offloading probability *for* k=3.

Fig. 4 also reveals that the values of p_2^{*VNF} and p_3^{*VNF} are higher than that of p_1^{*VNF}. This is due to the higher VNF service rates (c_2^{VNF} and c_3^{VNF}) compared to c_1^{VNF}. Additionally, the gap between p_2^{*VNF} and p_3^{*VNF} is relatively smaller, indicating that a significant increase in VNF capacity does not yield substantial improvements in the optimal probability of going to VNFs and average delay beyond a certain point. Therefore, it is important to choose the VNF service capacity appropriately.

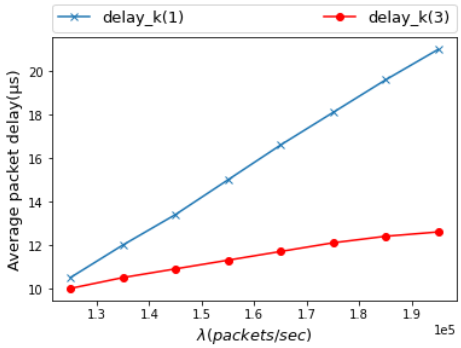

Fig. 5. Impact of arrival rate, λ on average packet delay for k = 1 and k = 3.

Additionally, in Fig. 5 we observe that the total average packet delay decreases as the number of VNFs used in the system increases. This is because offloading packets to multiple VNFs reduces the load on individual VNFs and the PNF, resulting in reduced packet delays. Interestingly, as the number of VNFs increases, the optimal probabilities for routing packets to each VNF (p_i^{*VNF}) shift toward a more balanced distribution. This balanced distribution allows for a more even workload distribution among different VNFs, leading to decreased load on each VNF and an overall improvement in system performance. However, it's essential to consider that a higher number of VNFs also introduces added complexity and costs to the system. Thus, maintaining a balance between system performance and cost becomes crucial.

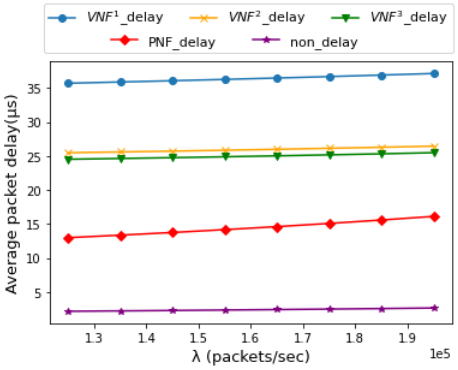

Fig. 6. Impact of arrival rate, λ on packet delay on different paths for k = 3.

In Fig. 6, we can see the impact of λ on the average packet delay for packets passing through all three VNFs (*VNF^1*, *VNF^2*, *and VNF^3*), *PNF*, and those that do not require any network function. Since our analytical result matches with our simulation result precisely, we present the graph for analytical analysis only. The delays for each type of packet are labeled as *VNF^1_delay*, *VNF^2_delay*, *VNF^3_delay*, *PNF_delay*, and *non_delay*. We can observe that as λ increases, packet delay increases for all types of packets. This is because the higher arrival rate leads to increased congestion in all the queues, resulting in more extensive delays. Notably, the slopes of packet delays differ as λ increases. The delay for packets that do not require any network function remains almost stable. On

the other hand, packet delays for VNF^1, VNF^2 and VNF^3 exhibit gradual increases, while the delay for PNF escalates at a faster rate. This discrepancy arises from the heavier load on the PNF queue compared to the queues of the VNFs. Multiple VNFs introduce additional propagation delay between the switch and VNFs, contributing to the higher delay observed in the PNF queue.

Once the packet arrival rate exceeds a certain point (160,000 pkts/sec), the delay for PNF increases notably due to the heavy load, while the delay for VNF increases only slightly due to the light load. Additionally, the delay for VNF^1 is much higher than that of VNF^2 and VNF^3 because the service rates of VNF^2 and VNF^3 are higher than that of VNF^1. The gap between VNF^2_delay and VNF^3_delay is smaller for the same reason.

The analytical result indicates that the packet delays for all three VNFs are significantly higher than that of PNF due to the longer path that packets requiring VNF must travel (as shown in (7) and (10)). This difference in delay also results from the additional propagation delay that packets experience between the switch and VNFs.

E. Impact of p^N

Fig. 7 shows the impact of the probability of requiring a network function (p^N) on both the average packet delay and the optimal offloading probability. As the percentage of packets requiring a network function (p^N) increases, the average packet delay also increases. This is because more packets require network functions, leading to higher loads on both the PNF and VNFs.

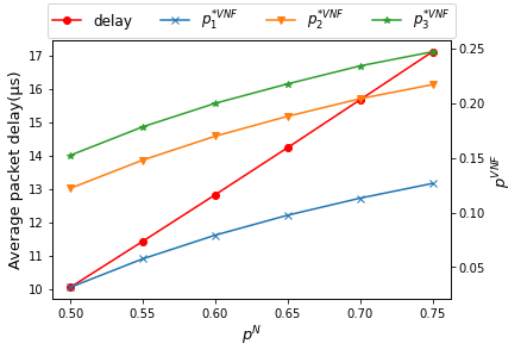

Fig. 7. Impact of p^N on delay and offloading probability for k = 3.

Additionally, we observe that as p^N increases, the optimal probabilities of packets offloaded to multiple VNFs (p_1^{*VNF}, p_2^{*VNF}, and p_3^{*VNF}) also increase. This is due to the fact that the service rate of the PNF (c^{PNF}) is much lower than that of the VNFs (c_1^{VNF}, c_2^{VNF}, and c_3^{VNF}) (as shown in Table III). As p^N increases, the average delay for packets requiring the PNF increases faster than packets requiring multiple VNFs. Therefore, offloading more packets to multiple VNFs helps to reduce the average packet delay. Furthermore, the probabilities of offloading to VNFs vary due to the differences in their service rates (c_1^{VNF}, c_2^{VNF}, and c_3^{VNF}), with p_2^{*VNF} and p_3^{*VNF} being greater than p_1^{*VNF}. Importantly, this difference is not affected by p^N as it is independent of this parameter. Therefore, the gap between the probabilities of going to VNFs (p_1^{*VNF}, p_2^{*VNF}, and p_3^{*VNF}) is fixed, regardless of the chosen p^N. This gap is solely attributed to the varying VNF service rates (c_1^{VNF}, c_2^{VNF}, and c_3^{VNF}).

F. Impact of c_1^{VNF}

Fig. 8 illustrates the impact of VNF service rate (c_1^{VNF}) on average packet delay and optimal offloading probabilities (p_1^{*VNF}, p_2^{*VNF}, and p_3^{*VNF}). As c_1^{VNF} increases, the packet delay for packets requiring multiple VNFs decreases, leading to an overall reduction in total average packet delay. Additionally, the increase in c_1^{VNF} initially raises all three optimal probabilities (p_1^{*VNF}, p_2^{*VNF} and p_3^{*VNF}), as it becomes more advantageous to direct more packets to multiple VNFs.

Increasing the capacity of c_1^{VNF} from 95,000 (pkts/sec) to 950,000 (pkts/sec) results in higher values for p_i^{*VNF}, and the gap between p_1^{*VNF} and p_2^{*VNF} decreases. Interestingly, when $c_1^{VNF} = c_2^{VNF}$ =950,000 pkts/sec, both VNFs have identical optimal probabilities for directing packets to VNFs, making p_1^{*VNF} equal to p_2^{*VNF}.

However, beyond a certain point, further increases in c_1^{VNF} result in an increase only in p_1^{*VNF}, while the values of p_2^{*VNF} and p_3^{*VNF} decrease. This phenomenon can be attributed to the fact that at higher values of c_1^{VNF}, the advantages of offloading more packets to VNF^1 become more beneficial, leading to a reduction in the optimal probabilities of routing packets to VNF^2 and VNF^3.

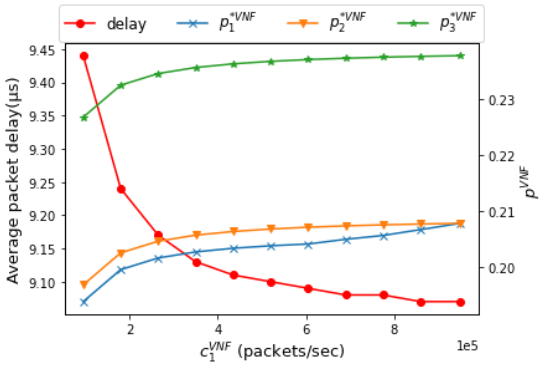

Fig. 8. Impact of c_1^{VNF} on delay and offloading probability for k = 3.

G. Impact of D_i^{SV}

Fig. 9 illustrates how increasing the fixed propagation delay (D_i^{SV}), impacts the average packet delay and the optimal offloading probabilities (p_1^{*VNF}, p_2^{*VNF}, p_3^{*VNF}). As D_i^{SV} increases, the average packet delay also increases because packets experience longer propagation delays to reach multiple VNFs. This phenomenon occurs because the benefits of offloading decrease as the fixed propagation delay becomes larger. Offloaded packets experience more delay due to the longer path they have to travel to reach multiple VNFs. As a result, the optimal offloading probabilities decrease to minimize the overall packet delay.

In summary, increase of fixed propagation delay results in higher average packet delays. It also decreases the optimal probabilities of offloading packets to VNFs. This information is crucial for network administrators and service providers to understand the trade-offs between fixed propagation delay, offloading probabilities, and overall system performance.

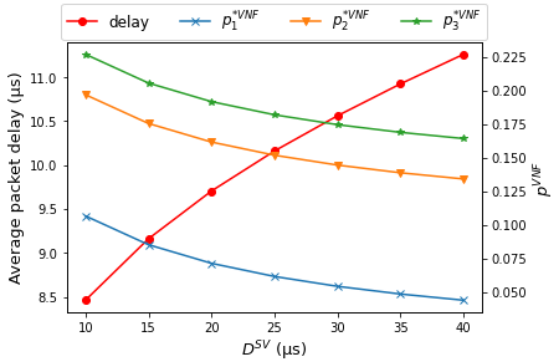

Fig. 9. Impact of D_i^{SV} on delay and offloading probability for k = 3.

As explained earlier, the optimal probabilities of offloading packets to VNFs (p_1^{*VNF}, p_2^{*VNF}, and p_3^{*VNF}) differ due to variations in VNF service rates (c_1^{VNF}, c_2^{VNF}, $and\ c_3^{VNF}$). Irrespective of the value of D_i^{SV}, this gap between probabilities remains constant because packet arrival rates and other parameters remain consistent. Thus, the gap is solely determined by the use of distinct VNF service rates and remains unaffected by changes in D_i^{SV}.

VI. Conclusion

In this paper, we have introduced an integration of multiple virtual network functions (VNFs) within a P4 switch, aiming to determine the optimal probabilities for directing packets to VNFs and ultimately minimizing the average packet delay. Our investigation encompassed various VNF computing capacities, allowing for a comprehensive comparison of their respective optimal offloading probabilities and their effects on performance metrics. To assess the average packet delay, we employed the M/M/1 queuing model. Also, we proposed an optimization solution based on gradient descent to find the optimal offloading probabilities for various VNF servers.

The study highlights the importance of VNF server computing capacities in determining the optimal offloading probabilities. Servers with larger computing capacities can process more packets, consequently yielding higher optimal offloading probabilities. Integrating more VNFs into the system reduces overall average packet delay. This is because by distributing packets among multiple VNFs, we can reduce workload on individual VNFs and the PNF. However, increasing the number of VNFs also increases system complexity and cost, which should be taken into consideration. The research findings demonstrate that optimal offloading from a P4 switch to three VNFs can yield reductions in average packet delay ranging from 4.76% to 40.02% when compared to a single VNF scenario. With the exception of D_i^{SV}, when other parameters (λ, p^N, c_i^{VNF}) increase, the average packet delay also increases, leading to the need for higher offloading probabilities. This trend aligns with the idea that offloading a greater number of packets to multiple VNFs can effectively mitigate packet delay. In conclusion, the study provides valuable insights for enhancing system performance, emphasizing the importance of considering VNF computing capacities and offloading probabilities.

This model provides a mathematical framework for analyzing multiple VNF queues with Poisson arrivals, exponentially distributed service times, and a single server. Recognizing the potential impact of different packet arrival distributions on results, we plan to explore alternative distributions in our future work.

References

[1] Q. Waseem, W. Isni Sofiah Wan Din, A. Aminuddin, M. Hussain Mohammed, and R. F. Alfa Aziza, "Software-Defined Networking (SDN): A Review," in 2022 5th International Conference on Information and Communications Technology (ICOIACT), 2022.

[2] S. Papavassiliou, "Software Defined Networking (SDN) and Network Function Virtualization (NFV)," Future Internet, vol. 12, no. 1, p. 7, 2020.

[3] H. A. Jawdhari and A. A. Abdullah, "The application of network Functions Virtualization on different networks, and its new applications in blockchain: A survey," Webology, vol. 18, no. Special04, pp. 1007–1044, 2021.

[4] H. Li et al., "Deployment of VNF service chains with grooming and resilience in elastic optical networks," Opt. Fiber Technol., vol. 81, no. 103482, p. 103482, 2023.

[5] B. Goswami, M. Kulkarni, and J. Paulose, "A survey on P4 challenges in software defined networks: P4 programming," IEEE Access, vol. 11, pp. 54373–54387, 2023.

[6] M. He, "P4NFV: An NFV Architecture with Flexible Data Plane Reconfiguration," in 14th International Conference on Network and Service Management (CNSM), IEEE, 2018, pp. 90–98.

[7] L. A. Makara, Y. C. Lai, Y. D. Lin, W. Seah, and A. Pekar, Offloading from P4 Switches to Nfv in Programmable Data Planes. Available at SSRN 4090265.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Van wijngaarden-dekker-brent method," in Numerical Recipes in FORTRAN: The Art of Scientific Computing, Cambridge, England: Cambridge University Press, 1992, pp. 352–355.

[9] J. Raychev, G. Hristov, D. Kinaneva, and P. Zahariev, "Modelling and evaluation of software defined network architecture based on queueing theory," in 28th EAEEIE Annual Conference (EAEEIE), Hafnarfjordur, Iceland, 2018.

[10] C. Sarkar and S. K. Setua, "Analytical model for OpenFlow-based software-defined network," in Advances in Intelligent Systems and Computing, Singapore: Springer Singapore, 2018, pp. 583–592.

[11] L. O. Nweke and S. D. Wolthusen, "Modelling adversarial flow in software-defined industrial control networks using a queueing network model," in IEEE Conference on Communications and Network Security (CNS), 2020.

[12] Y. Goto, B. Ng, W. K. G. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic OpenFlow–based switch model," Comput. Netw., vol. 164, no. 106892, p. 106892, 2019.

[13] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, and W. K. G. Seah, "Modelling Software-Defined Networking: Software and hardware switches," J. Netw. Comput. Appl., vol. 122, pp. 24–36, 2018.

[14] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, and W. K. G. Seah, "Full encapsulation or internal buffering in OpenFlow based hardware switches?," Comput. Netw., vol. 167, no. 107033, p. 107033, 2020.

[15] N. Pitaev, M. Falkner, A. Leivadeasy, and I. Lambadarisy, "Multi-VNF performance characterization for virtualized network functions," in 2017 IEEE Conference on Network Softwarization (NetSoft), 2017.

[16] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for VNF forwarding graph embedding," IEEE Trans. Netw. Serv. Manag., vol. 16, no. 1, pp. 98–112, 2019.

[17] D. Yamada and N. Shinomiya, "A solving method for computing and network resource minimization problem in service function chain against multiple VNF failures," in 2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC), 2019.

[18] S. Van Rossem, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "VNF Performance modelling: From stand-alone to chained topologies," Comput. Netw., vol. 181, no. 107428, p. 107428, 2020.

[19] G. Ramya and R. Manoharan, "Traffic-aware dynamic controller placement in SDN using NFV," J. Supercomput., vol. 79, no. 2, pp. 2082–2107, 2023.

[20] A. Fahmin, Y.-C. Lai, M. S. Hossain, and Y.-D. Lin, "Performance modeling and comparison of NFV integrated with SDN: Under or aside?," J. Netw. Comput. Appl., vol. 113, pp. 119–129, 2018.

[21] J. Billingsley, W. Miao, K. Li, G. Min, and N. Georgalas, "Performance analysis of SDN and NFV enabled mobile cloud computing," in GLOBECOM 2020-2020 IEEE Global Communications Conference, 2020.
[22] N. Surantha and N. A. Putra, "Integrated SDN-NFV 5G network performance and management-complexity evaluation," Future Internet, vol. 14, no. 12, p. 378, 2022.
[23] F. Paolucci, F. Cugini, P. Castoldi, and T. Osinski, "Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability," IEEE Netw., vol. 35, no. 3, pp. 154–160, 2021.
[24] S. Ji, DE4NF: High Performance Nfv Frameworkwith P4-Based Event System (Doctoral dissertation). 2020.
[25] T. Osinski, H. Tarasiuk, L. Rajewski, and E. Kowalczyk, "DPPx: A P4-based Data Plane Programmability and Exposure framework to enhance NFV services," in 2019 IEEE Conference on Network Softwarization (NetSoft), 2019.
[26] A. Mohammad Khan, S. Panda, S. G. Kulkarni, K. K. Ramakrishnan, and L. N. Bhuyan, "P4NFV: P4 enabled NFV systems with SmartNICs," in 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2019, pp. 1–7.
[27] T. Zhang, L. Linguaglossa, M. Gallo, P. Giaccone, L. Iannone, and J. Roberts, "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, 2019.
[28] G. I. Palmer, V. A. Knight, P. R. Harper, and A. L. Hawa, "Ciw: An open-source discrete event simulation library," J. Simul., vol. 13, no. 1, pp. 68–82, 2019.
[29] G. I. Palmer and Y. Tian, "Implementing hybrid simulations that integrate DES+ SD in Python," Journal of Simulation, pp. 1–17, 2021.
[30] H. Harkous, M. Jarschel, M. He, R. Pries, and W. Kellerer, "P8: P4 with predictable packet processing performance," IEEE Trans. Netw. Serv. Manag., vol. 18, no. 3, pp. 2846–2859, 2021.
[31] S.-Y. Wang, J.-Y. Li, and Y.-B. Lin, "Aggregating and disaggregating packets with various sizes of payload in P4 switches at 100 Gbps line rate," J. Netw. Comput. Appl., vol. 165, no. 102676, p. 102676, 2020.

Farhin Faiza Neha is pursuing an MSc in Computer Science and Engineering at Bangladesh University of Engineering and Technology (BUET). She has completed BSc 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 PhD, 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.