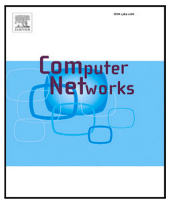
Computer Networks 257 (2025) 110907

Contents lists available at ScienceDirect

Computer Networks

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



P4+NFV: Optimal offloading from P4 switches to NFV for diverse traffic streams

Sidharth Sharma [a,*], Yuan-Cheng Lai [b], Ashwin Gumaste [c], Ying-Dar Lin [d]

[a] Department of Computer Science and Engineering, Indian Institute of Technology Indore, Indore, 453552, India
[b] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[c] Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai, 400076, India
[d] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Keywords:
P4
NFV
Offloading
Optimization
Programmability
Data plane

ABSTRACT

Software-defined Networking (SDN) is making its mark in the operator networks. The latest generation of SDN switches supporting paradigms such as P4, are paving the way for complete data plane programmability. Though P4 switches enable some newer applications, they do not provide the same agility and scalability offered through fully programmable softwarized data planes of Virtual Network Functions (VNFs). This paper argues that to achieve substantial performance benefits, an operator can take advantage of deploying P4 switches alongside VNFs. The idea is to use the P4 switch as a default packet handler, whereas traffic can be offloaded to an off-site VNF when queues at the switch start to build. To this end, this paper first models the queuing behavior of a networked system with a P4 switch and a VNF to determine the queuing delay induced by such a hybrid architecture. While doing so, the paper considers two different cases of homogeneous and heterogeneous traffic patterns. Subsequently, the paper proposes algorithms for finding optimal traffic offloading, leading to overall delay minimization. The paper showcases significant performance gains of optimally offloading traffic from a P4 switch to a VNF amidst changes in disparate traffic and network parameters through simulations and analytical results. For instance, at moderate loads of homogeneous traffic, optimal offloading yields performance gains of up to 76.44% over a scenario where a P4 switch handles all the packets. For heterogeneous traffic patterns, the results show that the dominant flow's workload and average packet size can significantly impact the offloading performance.

1. Introduction

Complex network functions and a diverse set of protocols will characterize modern applications such as 5G, IoT, and beyond. To realize these next-generation services in a network, data planes should ideally be fully programmable [1,2]. Software-defined Networking (SDN) was the initial step towards making the networks programmable by decoupling the control plane from the data plane. However, forwarding devices supporting SDN protocols (such as OpenFlow) are inadequate for next-generation applications as they only support partial data plane programmability [3]. This is because OpenFlow cannot program the packet processing pipeline at the forwarding devices. In the absence of full programmability in OpenFlow-based SDN switches, proprietary hardware middleboxes still needed to be deployed for facilitating new applications [4]. However, they incur huge costs and require operational expertise to manage.

Network Function Virtualization (NFV) is a recent technological advancement towards achieving full programmability in the networks by detaching the network function from the customized hardware [5,6]. Instead, the software implementations of the network functions (referred to as virtual network functions or VNFs) are hosted on commodity servers leading to significant cost benefits and flexibility [6].

Recent developments in the capability to customize the switch's data plane through high-level programming languages (such as P4) enable significant control and flexibility in packet processing [7]. P4 makes OpenFlow supporting SDN switches fully programmable, which can also be used to realize heterogeneous network functions (such as NAT, Firewall, Load Balancer, DNS Accelerator, BNG), generally implemented in proprietary middleboxes [8,9]. Traditionally, for traversing Network Functions (NFs), traffic is redirected from the operator's switches towards the middleboxes [5]. Replacing these general-purpose switches with their P4-based programmable equivalents yields significant performance benefits. This is because the traffic now need not traverse a remote middlebox as the programmable switch can host

* Corresponding author.
E-mail addresses: sidharth@iiti.ac.in (S. Sharma), laiyc@cs.ntust.edu.tw (Y.-C. Lai), ashwing@ieee.org (A. Gumaste), ydlin@cs.nctu.edu.tw (Y.-D. Lin).

https://doi.org/10.1016/j.comnet.2024.110907
Received 9 October 2023; Received in revised form 31 July 2024; Accepted 6 November 2024
Available online 14 November 2024

the Physical Network Function (PNF). Moreover, these programmable switches could be placed at the network's edge, facilitating edge computing [10]. In such a scenario, implementing the PNFs within these switches makes more sense for latency-sensitive applications. However, resources at the programmable switches are distributed across multiple operations; the network function gets only a chunk of the switch's computational resources, making the PNF implementation restrictive in terms of capacities for processing network functions. Also, some network functions are complex; the P4 switch must spend a lot of computation capacity to handle the packets that require these network functions. Compared to commodity servers with gigabytes of memory that can easily host NFs, programmable data-plane switches typically only have SRAMs and TCAMs in the order of megabytes [11]. For instance, the Intel Tofino3 switch has 200 Megabits of SRAM and 10.3 Megabits of TCAM per pipeline [11]. Therefore, if the switch's NF cannot process the amount of traffic it receives, the edge computing paradigm suggests offloading traffic to a cloud (or Data Centers (DCs)) that offers substantial processing capacities to deal with a tremendous amount of network traffic. For hosting NFs, DCs are assumed to have VNF farms, which are extremely flexible in terms of capacity as they can be scaled/replicated as per the traffic demand. However, to traverse VNFs, the network traffic must be diverted towards the operator's DC from the switch on the traffic path, leading to an additional delay penalty for the traffic. Therefore, both the software and P4-based network functions have their benefits, but their adoption has some challenges.

In this paper, we evaluate the performance benefits of using the two alternatives together. We propose using a P4-based programmable switch as a default path while traffic is offloaded to a VNF if needed. The offloading depends upon the traffic volume at the programmable switch. Through an analytical model, we show that significant performance benefits can be obtained using the proposed model instead of entirely using hardware- or software-based NFs.

Recently, researchers investigated the possibility of cooperating a programmable P4 switch with NFV. P4-SC [12] and P4-SFC [13] are methods of implementing VNF service function chains with a P4-capable device. P4NFV [14] and Hyper [15] are proposals for running NFs on both software and hardware platforms to improve performance. UNO [16] is a SDN-controlled NF offload architecture that can transparently leverage the smart NIC's programmable compute capabilities to accelerate the NF data plane. In [17] authors proposed an architecture for embedding P4 capability into NFV platforms, including both the host and smart NIC. In [18], is a P4-based framework to enhance NFV services. The framework presents a way to offload traffic utilizing P4 programs. Though these works explored the interoperability of NFV and programmable switches, none considered the performance gain from their optimal usage together. We argue that even though the integration of programmable switches with NFV appears promising, significant performance improvement can only be observed if the offloading is optimal. We answer multiple questions, which could be helpful for an operator attempting to integrate programmable switches with VNFs. These questions include the impact of parameters (such as arrival rate, the distance between the programmable switch and the VNF, packet attributes, and processing resources) on the optimal offloading and average packet delay.

In the paper, we present two different cases of traffic arrivals at the programmable switch. The first case is when the traffic arriving at the switch is of the same class and can be characterized by a single arrival rate. In contrast, the second case considers different classes of traffic with independent arrival rates. The second class is useful for operators who offer different heterogeneous traffic types and their arrival rates are independent of each other.

The paper makes the following contributions:

- We, for the first time, discuss the possibility of offloading traffic from a programmable switch to a VNF. To analyze possible performance gains, we use queuing theory to obtain the average packet delay.
- We consider two cases of homogeneous and heterogeneous traffic and for both cases proposed algorithms for optimal offloading of traffic from a programmable switch to a VNF, aiming to minimize the average packet delay.
- With the evaluation, we answer multiple questions, which could be helpful for an operator attempting to integrate programmable switches with VNFs. These questions include the impact of parameters (such as arrival rate, the distance between the programmable switch and the VNF, packet attributes, and processing resources) on the optimal offloading and average packet delay.

The paper is organized as follows. Section 2 presents some background on P4 and NFV and discusses a few works related to our performance modeling. Section 3 first proposes the architecture of the programmable switch with NFV and then analyzes its performance. Section 4 describes algorithms for optimal VNF offloading amidst homogeneous and heterogeneous traffic patterns. Section 5 showcases results from the analytical and simulation models. Finally, Section 6 concludes the paper.

2. Background and related work

Data plane programmability offers infrastructural support for next-generation services. Programming Protocol-independent Packet Processors (P4) is a de-facto language for programming a data plane. P4 can dynamically parse any header field from the packet and pass it through a custom pipeline of match-action tables. With P4, new applications can be quickly prototyped by network vendors or even users, eliminating the reliance on Application-specific integrated circuits (ASIC). This is the reason why operators across the globe have now started using P4 switches in their networks for different applications. Network functions like load balancers, DNS caches, tunnel gateways, firewalls, and DDoS detectors are already being implemented in the P4 switches [19,20]. In [8] authors have presented a DNS service within a network device using P4. Latency and throughput improvements over existing software solutions were shown through experiments. In [21], authors proposed a method for offloading media traffic from relay servers to P4 programmable switches. P4 switch vendors look at IoT and 5G as significant opportunities, hence designing high-speed programmable switches without compromising performance [22].

NFV enables disaggregation of network functions from the proprietary middleboxes. Like the programmable switches, NFV also introduces programmability in the network and facilitates many new applications. To our knowledge, this is the first work that together analyzes the performance of a P4-based programmable switch with NFV. However, some previous papers analyzed the SDN/NFV networks. We list some major related works in Table 1 and qualitatively compare our work with them. We have categorized their performance modeling approach into approximate and exact analysis classes. In the approximate analysis class, [23–26,29–31,42] used M/M/1 or its variants for performance modeling of SDN switches. Whereas, [27,28,32] used M/G/1 model, M/Geo/1 and MMPP/M/1 model, respectively. In contrast, several works performed an exact analysis of SDN switches such as [36–38,40,41]. These works have used Markov chains for modeling the SDN switch's performance.

Authors in [23] analyzed the performance of a single OpenFlow SDN switch and controller. In [24] is also an M/M/1 analysis of an SDN switch with an OpenFlow controller using Jackson networks. However, their analysis can be extended for more than one switch. In [26] is an approach to analyze how OpenFlow packet-in messages impact the SDN switch and controller performance using the M/M/1 model. The work in [25] is focused on the SDN data plane performance and proposed a preemption-based packet scheduling approach. The proposed approach enhances the global fairness index and reduces packet loss probability. The work of [27] also solely considered SDN data plane performance and used an M/Geo/1 queuing model to obtain the delay values.

Table 1
Related works.

Category	Paper	SDN switch	NF		Characteristics
			VNF	PNF	
Approximate	[23] Jarschel (2011)	M/M/1	No	No	Fundamental modeling
	[24] Mahmood (2015)	M/M/1	No	No	Multiple switches
	[25] Miao (2015)	HPQ:M/M/1/k LPQ:M/M/1/k	No	No	Priority queues
	[26] Shang (2016)	M/H2/1	No	No	Separation of packet-in messages and others
	[27] Sood (2016)	M/Geo/1	No	No	Geometric distribution for service
	[28] Miao (2016)	HPQ:MMPP/M/1 LPQ:MMPP/M/1/k	No	No	MMPP for multimedia traffic arrivals
	[29] Xiong (2016)	Mx/M/1	No	No	Batch traffic arrival rate at switch
	[30] Fahmin (2018)	M/M/1	M/M/1	No	Combination of SDN and VNF
	[31] Nweke (2020)	M/M/1	No	No	Adversarial flow
	[32] Zhao (2020)	M/G/1	No	No	Software-defined WAN
	[33] Shen (2022)	M/M/1	No	No	Data Center Network
	[32] Zhao (2020)	M/G/1	No	No	Software-defined WAN
	[33] Shen (2022)	M/M/1	No	No	Data Center Network
	[34] Maheswari (2022)	M/M/c	No	No	Multi-Controller SDN
	[35] Kröger (2022)	M/G/1	No	No	P4 switch
	This paper	**M/M/1 M/G/1**	**M/M/1 M/G/1**	**M/M/1 M/G/1**	**P4 switch with VNF**
Exact	[36] Goto (2019)	2D MC (HPQ, LPQ)	No	No	Exact solution with Markov Chain
	[37] Singh (2018)	2D MC (HPQ, LPQ)	No	No	Software vs. hardware switches
	[38] Lai (2019)	3D MC (flow1, flow2, Q)	No	No	Flow-level: TCP and UDP
	[39] Zhang (2019)	2D MC (Q1, Q2)	No	No	Auxiliary connection between switch and controller
	[40] Singh (2020)	4D MC (internal buffer, HPQ, LPQ, hardware)	No	No	Encapsulation vs. Internal buffer
	[41] Gadallah (2022)	7D MC (queues and flows (TCP/UDP) at switch, local and root controllers)	No	No	Flow-level TCP and UDP

In [28], authors used Markov Modulated Poisson Process (MMPP) to model bursty and correlated arrivals for an SDN switch. Whereas the work in [32] targets performance modeling of a software-defined WAN. They have used M/M/n queue to model the controller cluster and an M/G/1 queue for the OpenFlow switch. Authors in [33] propose a queuing model for estimating the flow table states of SDN switches in DCNs. In another paper [34], authors considered c controllers with finite buffers of size k and evaluated the effect of Retention of Reneged flows. The authors have used M/M/c/K queuing model in their analytical framework. The work in [35] is close to our work in the sense that they model the performance of a P4 switch using M/G/1 queues. However, the authors have not considered VNF in their model. The work in [43] uses deterministic network calculus for obtaining delay and buffer sizes for the SDN switch and controller.

On the exact analysis front, the work in [36] uses priority queues with a finite capacity for classful treatment of packets arrived at an SDN switch. They have used a continuous-time Markov chain (CTMC) to model the system. The work in [37] presents an analysis of hardware- and software-based SDN switches using a Quasi-Birth Death (QBD) process. In [40], authors evaluate two schemes for the SDN switch and controller communication. For both methods, the authors analyzed the resultant performance using CTMC. Whereas the work in [38] analyzed TCP and UDP flows in SDN switches. Their analysis also captured the communication between the SDN switch and controller using CTMC. Similarly, presented in [39] is a queuing analysis of auxiliary-connection-enabled OpenFlow switches using CTMC. In [41], authors have used a 7-D state to model TCP and UDP flows in an SDN environment. The authors have proposed a queuing model for SDN traffic considering TCP and UDP flows in a network with multiple controllers and OF switches. All the queues were modeled as M/M/1/m queues.

Moreover, a few works characterize the performance of NFV networks. Notably, [44] used stochastic network calculus to analyze the performance of NFV servers as part of a service function chain. In [45] is an end-to-end performance modeling approach for a system consisting of Multi-access Edge Computing servers and VNFs using discrete-time Markov chains (DTMC). In [46], a network calculus-based analysis for a service function chain in an NFV network is presented. Authors in [47] proposed an M/D/1 queuing analysis of a VNF chain in a 5G core network. A dominant-resource generalized processor sharing (DR-GPS) approach is used for resource sharing among flows on an NFV node.

Approaches mentioned above have either considered SDN or NFV networks in their analysis. Though some papers consider the intricacies of operating them together, such as [30,48]. Authors in [30] modeled a VNF along with an SDN switch and controller. They have considered two scenarios based on whether VNF can directly interact with the switch or the SDN controller. They used a network of M/M/1 queues to model all the considered elements and derived delay bounds for both cases. The work in [48] derives the performance of a mobile cloud computing platform amidst the presence of both SDN and NFV. Like [30,48] has also used M/M/1 queues in their analysis.

Our work is fundamentally different from all the related works, as none considered analyzing the performance of a P4 switch along with a VNF. After analyzing the queuing network, we aim to find the optimal offloading probability for a packet leading to overall delay minimization.

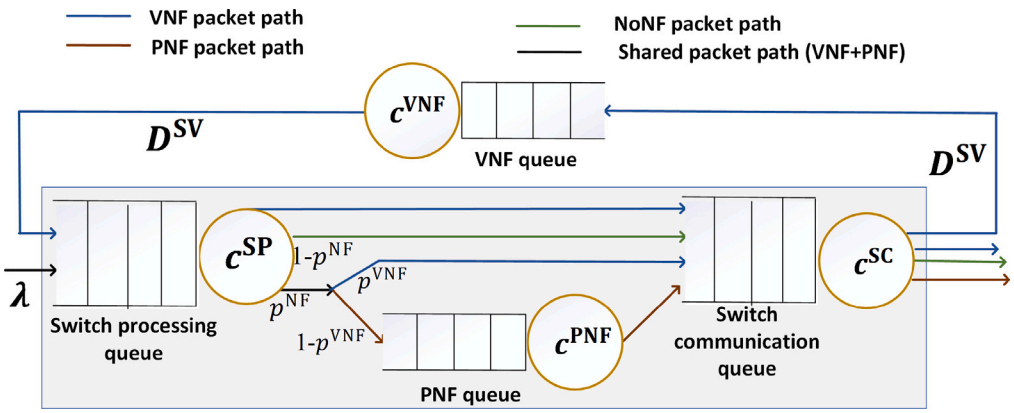


Fig. 1. Queuing representation of a programmable switch with a VNF for homogeneous traffic.

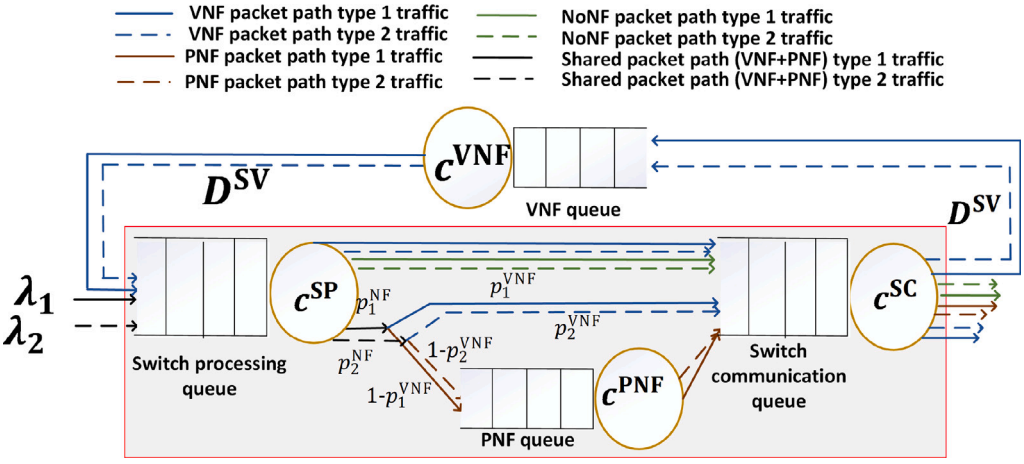


Fig. 2. Queuing representation of a programmable switch with a VNF for heterogeneous traffic.

3. Problem formulation

We first briefly explain the network architecture used for developing the analytical model. We assume that the operator has deployed programmable switches instead of traditional ones. The programmable switch is assumed to be placed at the edge of the data center where VNF resides. Therefore, traffic entering a switch requiring a network function can either traverse the function in the switch or can be forwarded towards the VNF in the DC. However, after VNF traversal, the packets re-enter the switch and eventually be forwarded towards their destination. In this scenario, the critical question is how much traffic should be offloaded to the VNF so that the operator gets the maximum performance benefits. This section first presents a queuing network emulating the discussed architecture. Based on this network, the average packet delay is derived. Finally, the problem of estimating optimal offloading is formulated. We consider two different scenarios in this paper. The first one is for homogeneous traffic, where we consider one stream of traffic and calculate offloading probability. For the second scenario, we consider multiple heterogeneous traffic streams to be handled differently by the switch. For example, some of these traffic streams may require a specific network function, while others may not require traversing it.

3.1. Model

We model the interoperability of a programmable switch with a VNF using interconnected queues. Figs. 1 and 2, which correspond to homogeneous and heterogeneous traffic, respectively, are the queuing networks considered in the paper. As shown in both the figures, there are four queues in our model: (a) A processing queue that serves every packet that enters the switch; a PNF queue at the switch that handles the packet traversing the network function inside the switch; a communication queue that handles every packet transmitted out of the switch; and finally, a queue at the VNF that handles packets forwarded to the VNF from the switch.

There are some assumptions we follow in our analysis. We assume all the queues are of infinite length. The packet arrival at the switch follows the Poisson distribution. For homogeneous traffic, the service times at each queue are exponentially distributed. While for the heterogeneous case, the service times are assumed to be generally distributed. We also assume that a packet can traverse the NF at most once, and there is no separation of newly coming packets and the packets which have experienced VNF in the input queue. Our final assumption is about the VNF queue, which is assumed to be the queue entering the VNF farm of the DC.

Table 2
Notations.

λ	Input packet arrival rate
p^{VNF}	Switch to VNF offloading probability
p^{NF}	Probability of traversing the network function
c^{PNF}	PNF processing capacity
c^{VNF}	VNF processing capacity
c^{SP}	Switch processing capacity
c^{SC}	Switch communication capacity
w	Packet workload
l	Packet length
D^{SV}	Propagation delay between the switch and VNF

It is assumed that the offloading algorithm runs on the P4 switch itself. Our algorithm takes less than 0.5 s to find the optimal offloading probability. The switch keeps track of the arrival rate of the flows and runs the algorithm after every 0.5 s, assuming that the arrival rate will be similar for the next 0.5 s. Based on the algorithm's output, the switch offloads the traffic to the VNF.

3.2. Delay modeling for homogeneous traffic

We now derive the average packet delay experienced by the packets in the considered queuing network for the homogeneous traffic case (shown in Fig. 1). In this regard, we first calculate the cumulative arrival rate at each of the four queues shown in Fig. 1 and then use M/M/1 formulae for calculating the average delay. Notations used in this analysis are presented in Table 2.

3.2.1. Calculating delay at switch's processing queue

Initially, when packets with an arrival rate of λ enter the switch, they are handled by the switch's processing queue. Other than these newly arrived packets, the packets offloaded to the VNF from the switch re-enter it and are handled by the switch's processing queue. Therefore, the cumulative arrival rate at the switch's processing queue is calculated as

$$\lambda^{\mathrm{SP}} = \lambda + \lambda \times p^{\mathrm{NF}} \times p^{\mathrm{VNF}}. \tag{1}$$

Since we use M/M/1 queuing model, the average packet delay at the switch's processing queue (L^{SP}) can be obtained as

$$L^{\mathrm{SP}} = \frac{1}{c^{\mathrm{SP}} - \lambda^{\mathrm{SP}}}. \tag{2}$$

3.2.2. Calculating delay at switch's PNF queue

Out of the packets requiring network function traversal, some are handled by the PNF at the switch. Hence arrival rate at the switch's PNF queue is

$$\lambda^{\mathrm{PNF}} = \lambda \times p^{\mathrm{NF}} \times (1 - p^{\mathrm{VNF}}). \tag{3}$$

The processing capacity at the switch's PNF queue is c^{PNF} and the average packet workload is w. Hence the packet service rate at the PNF's queue is $\mu^{\mathrm{PNF}} = c^{\mathrm{PNF}}/w$. Therefore, the average delay at the PNF queue (L^{PNF}) is

$$L^{\mathrm{PNF}} = \frac{1}{\mu^{\mathrm{PNF}} - \lambda^{\mathrm{PNF}}}. \tag{4}$$

3.2.3. Calculating delay at switch's communication queue

Any packet supposes to leave the switch enters into its communication queue for transmission. Hence the arrival rate at the switch's communication queue is calculated as

$$\lambda^{\mathrm{SC}} = \lambda + \lambda \times p^{\mathrm{NF}} \times p^{\mathrm{VNF}}. \tag{5}$$

The capacity of the switch's communication queue is c^{SC}, and the average packet length is l. Hence the service rate at the switch's communication queue is $\mu^{\mathrm{SC}} = c^{\mathrm{SC}}/l$. Therefore, the average packet delay at the communication queue (L^{SC}) can be expressed as

$$L^{\mathrm{SC}} = \frac{1}{\mu^{\mathrm{SC}} - \lambda^{\mathrm{SC}}}. \tag{6}$$

3.2.4. Calculating delay at VNF queue

Some of the traffic entering the switch is forwarded towards the VNF. Hence the arrival rate at the VNF is

$$\lambda^{\mathrm{VNF}} = \lambda \times p^{\mathrm{NF}} \times p^{\mathrm{VNF}}. \tag{7}$$

The processing capacity of the VNF is c^{VNF} and the average packet workload is w. Hence the packet service rate at the VNF's queue is $\mu^{\mathrm{VNF}} = c^{\mathrm{VNF}}/w$. Therefore, the average packet delay at the VNF queue (L^{VNF}) is

$$L^{\mathrm{VNF}} = \frac{1}{\mu^{\mathrm{VNF}} - \lambda^{\mathrm{VNF}}}. \tag{8}$$

3.3. Delay modeling for heterogeneous traffic

For heterogeneous traffic, the queuing network is shown in Fig. 2. The network is approximated with M/G/1 queues. We assume there are N independent traffic classes in the network and for each class i its arrival rate (λ_i) and probability of requiring the network function traversal (p_i^{NF}) is known. The offloading probability (p_i^{VNF}) could also be different for each class i and depends upon each other. The cumulative arrival rates for switch processing queue, PNF queue, switch communication queue and VNF queue are calculated as follows

$$\lambda^{\mathrm{SP}} = \sum_{i=1}^{N} \lambda_i + \lambda_i \times p_i^{\mathrm{NF}} \times p_i^{\mathrm{VNF}}, \tag{9}$$

$$\lambda^{\mathrm{PNF}} = \sum_{i=1}^{N} \lambda_i \times p_i^{\mathrm{NF}} \times (1 - p_i^{\mathrm{VNF}}), \tag{10}$$

$$\lambda^{\mathrm{SC}} = \sum_{i=1}^{N} \lambda_i + \lambda_i \times p_i^{\mathrm{NF}} \times p_i^{\mathrm{VNF}}, \tag{11}$$

$$\lambda^{\mathrm{VNF}} = \sum_{i=1}^{N} \lambda_i \times p_i^{\mathrm{NF}} \times p_i^{\mathrm{VNF}}. \tag{12}$$

3.3.1. Calculating delay at switch's processing queue

Switch processing delay calculation is same as homogeneous traffic model and the queue can be approximated as M/M/1 queue. Hence delay at the processing queue can be calculated as

$$L^{\mathrm{SP}} = \frac{1}{c^{\mathrm{SP}} - \lambda^{\mathrm{SP}}}. \tag{13}$$

3.3.2. Calculating delay at switch's communication queue

For calculating delay at the switch's communication queue, we first need to calculate mean ($E[l]$) and variance of service time ($E[l^2]$) by using the following equations

$$E[l] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \frac{\lambda_i \times l_i}{c^{\mathrm{SC}}}, \tag{14}$$

$$E[l^2] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i \times \left(\frac{l_i}{c^{\mathrm{SC}}}\right)^2. \tag{15}$$

Now, we can use M/G/1 delay formula for calculating delay at switch's communication queue as

$$L^{\mathrm{SC}} = \frac{E[l^2] \times \lambda^{\mathrm{SC}}}{2 \times (1 - (E[l] \times \lambda^{\mathrm{SC}}))}. \tag{16}$$

3.3.3. Calculating delay at switch's PNF queue

For the PNF queue, mean ($E[w]$) and variance of service time ($E[w^2]$) can be calculated as

$$E[w] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \frac{\lambda_i \times w_i}{c^{\text{PNF}}}, \tag{17}$$

$$E[w^2] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i \times \left(\frac{w_i}{c^{\text{PNF}}}\right)^2. \tag{18}$$

By using M/G/1 delay formula, the average delay can be calculated as

$$L^{\text{PNF}} = \frac{E[w^2] \times \lambda^{\text{PNF}}}{2 \times (1 - (E[w] \times \lambda^{\text{PNF}}))}. \tag{19}$$

3.3.4. Calculating delay at VNF queue

Similarly, for the VNF queue, mean ($E[w]$) and variance of service time ($E[w^2]$) can be calculated as

$$E[w] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \frac{\lambda_i \times w_i}{c^{\text{VNF}}}, \tag{20}$$

$$E[w^2] = \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i \times \left(\frac{w_i}{c^{\text{VNF}}}\right)^2. \tag{21}$$

Now, delay can be calculated similarly to the PNF queue by using the following equation

$$L^{\text{VNF}} = \frac{E[w^2] \times \lambda^{\text{VNF}}}{2 \times (1 - (E[w] \times \lambda^{\text{VNF}}))}. \tag{22}$$

C. Calculating the Average Packet Delay: A packet, when enters a programmable switch, can follow one of the following three paths:

Path 1: If a packet does not require to traverse a network function, it can be directly sent towards its destination after traversing the processing and communication queue of the switch. For such packets, average packet delay can be calculated as

$$D^{\text{NoNF}} = L^{\text{SP}} + L^{\text{SC}}. \tag{23}$$

Path 2: If a packet is required to traverse the network function within the switch, it traverses the processing queue, PNF queue, and communication queue. Therefore, the average delay of packets traversing the PNF can be calculated as

$$D^{\text{PNF}} = L^{\text{SP}} + L^{\text{SC}} + L^{\text{PNF}}. \tag{24}$$

Path 3: If a packet requiring the network function is offloaded to a VNF, then after traversing the processing and communication queue, it is forwarded towards the VNF. After VNF traversal, it re-enters the switch's processing queue. Finally, it exits the switch after traversing its communication queue. The average delay for the packets traversing the VNF can be calculated as

$$D^{\text{VNF}} = 2 \times L^{\text{SP}} + 2 \times L^{\text{SC}} + 2 \times D^{\text{SV}} + L^{\text{VNF}}, \tag{25}$$

where D^{SV} is the propagation delay between the switch and VNF. Now, we can calculate the average packet delay for all the packets entering the switch based on the probability of packets choosing one of these three paths. The average packet delay for the homogeneous case can be calculated as

$$\begin{aligned} Delay_{\text{homo}} &= p^{\text{NF}} \times p^{\text{VNF}} \times D^{\text{VNF}} + \\ & p^{\text{NF}} \times (1 - p^{\text{VNF}}) \times D^{\text{PNF}} + (1 - p^{\text{NF}}) \times D^{\text{NoNF}}. \end{aligned} \tag{26}$$

For the heterogeneous traffic case, the average delay is calculated as

$$\begin{aligned} Delay_{\text{het}} &= \frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i (p_i^{\text{NF}} \times p_i^{\text{VNF}} \times D^{\text{VNF}} + \\ & p_i^{\text{NF}} \times (1 - p_i^{\text{VNF}}) \times D^{\text{PNF}} + (1 - p_i^{\text{NF}}) \times D^{\text{NoNF}}). \end{aligned} \tag{27}$$

3.4. Problem statement

After calculating the average packet delay, we are now in a position to explain our main problem of finding the optimal offloading probability. We define the optimal offloading probability as a value that results in the minimum average packet delay. This means, given the traffic arrival rate, probability of network function requirement, processing capacities (of the switch, PNF, and VNF), communication capacity of the switch, packet length and packet workload, we want to know the optimal offloading probability that minimizes the average packet delay. Formally, given the values of λ, p^{NF}, c^{PNF}, c^{SP}, c^{VNF}, c^{SC}, w, l and D^{SV}, we attempt to find the optimal p^{VNF} that minimizes the average packet delay obtained from Eq. (26) for homogeneous traffic and Eq. (27) for heterogeneous traffic. VNF offloading could offer delay improvements, but at the same time, we must consider the communication cost between the switch and the VNF to analyze the computation–communication trade-off.

To find the optimal p^{VNF} for the homogeneous traffic case, we make use of a space-search algorithm discussed in Section 4.1. For heterogeneous traffic, the problem of finding optimal offloading probability is slightly complex. This is because, for each traffic type, the offloading probability would be different, leading to the overall minimization of average packet delay across the traffic types. Therefore, the algorithm proposed for the homogeneous traffic scenario will not work. We formulate a constrained optimization model for finding optimal offloading probability in the heterogeneous traffic case. The objective of the optimization model is to minimize the overall delay and expressed as follows

$$\begin{aligned} &\text{Minimize} \quad Delay_{\text{het}}, \\ &\text{subject to} \quad 0 < p_i^{\text{VNF}} < 1, \quad \forall i = 1 \text{ to } N. \end{aligned} \tag{28}$$

The approach to solving this optimization is presented in Section 4.2.

Algorithm 1: Algorithm for finding optimal p^{VNF}

```
input: precision, initProb, reductionFactor
stepSize = initProb / reductionFactor
p^VNF = initProb
while (stepSize ≥ precision): do
    p^VNF = minSearch (stepSize,
    max[0, p^VNF - (reductionFactor × stepSize)],
    min[1, p^VNF + (reductionFactor × stepSize)])
    stepSize /= reductionFactor
return p^VNF
```

Algorithm 2: $minSearch(stepSize, min, max)$

```
result = min
minVal = ∞
x = min
while x ≤ max: do
    val = calcLatencyforProb(x)
    if (val < minVal): then
        minVal = val
        result = x
    x += stepSize
return result
```

4. Finding optimal offloading probability

4.1. Homogeneous traffic

As discussed in the previous section, we use a space-search algorithm for finding optimal offloading probability in the homogeneous traffic case. The proposed algorithm intelligently searches the state space and returns a p^{VNF} value that minimizes the average packet delay.

The algorithm has two procedures. Algorithm 1 is the main procedure, while Algorithm 2 is a procedure used by Algorithm 1. In

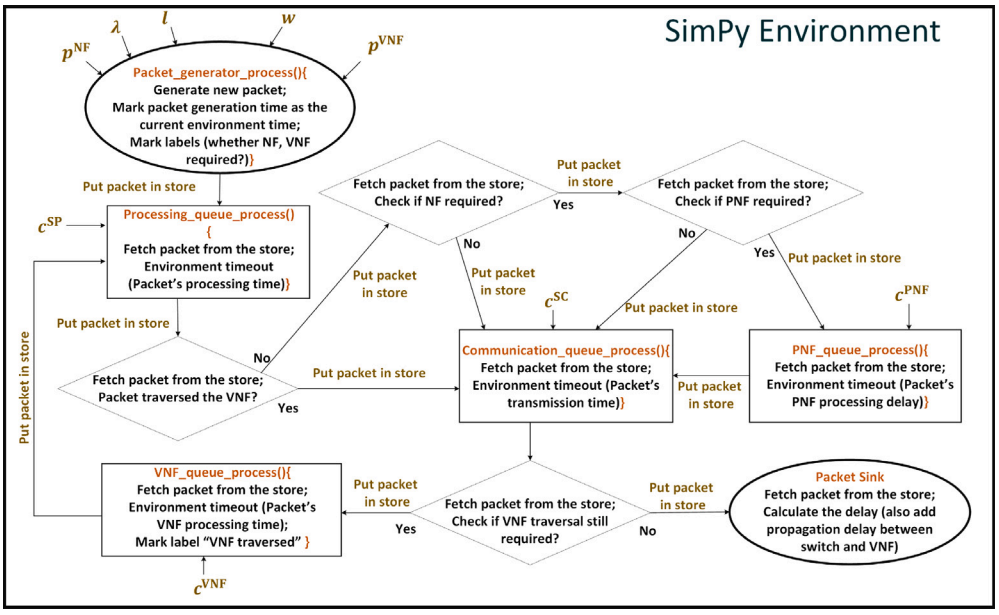


Fig. 3. Components of the simulator.

Algorithm 1, an input parameter *precision* represents the required precision while obtaining the optimal p^{VNF} value. Another parameter *reductionFactor* is used to narrow down the step size while proceeding towards the solution. Algorithm 1 initializes the initial probability (*initProb*) to 0.5 and also sets *precision* to a minimal value of $1e-6$ and *reductionFactor* to 10. The algorithm initializes the p^{VNF} value with *initProb* and starts searching for the optimal around it (in both directions) by creating a search space in every step. In every step, Algorithm 1 uses Algorithm 2. For each value in the search space, Algorithm 2 utilizes the average packet delay formula (Eq. (26)) and obtains the delay. Note that the function $calcLatencyforProb(x)$ returns the average packet delay at a p^{VNF} value of x. Other inputs to this function are the parameters used in the queuing analysis (explained in Table 2). Algorithm 2 returns the p^{VNF} corresponds to the lowest delay obtained among all the values in the search space. Based on the p^{VNF} value obtained in every step, Algorithm 1 modifies the search space. Also, after each step, Algorithm 1 narrows down the search space using *reductionFactor* and eventually identifies the optimal p^{VNF}.

4.2. Heterogeneous traffic

We need to solve the optimization model presented in Eq. (28) for the heterogeneous traffic case. The formulation is non-convex because of the non-convexity of the delay formulae in the objective function. Because of the non-convexity, there will be multiple extrema. Hence a method of choice to solve this model should be able to escape from local minima.

Simulated annealing is a widely used technique to find global minima in the presence of multiple local minima. The word *annealing* comes from a metallurgical process of slow and controlled cooling to make more robust materials. This process is a step-wise operation, where temperature changes in every step. The first step of the process starts by choosing an initial configuration and an initial temperature value applied to it. Every configuration has an associated energy value (denoted by E) that implies the goodness of the configuration. In every step, the initial configuration is altered using a visiting distribution function (a move that alters the existing configuration), and a new configuration is found. Finally, at the end of the step, the energy difference (ΔE) is calculated, which is the difference between the energy of the initial configuration and the new configurations' energy. If ΔE is positive, the new configuration is straight-away accepted. Otherwise, the new configuration is accepted based on an acceptance probability. The temperature is decreased for the next steps, and the same procedure is repeated until the global optima is found. As the temperature decreases, the probability of accepting bad moves (which are in the opposite direction of the global optima) is also minimized.

Classical simulated annealing (CSA) [49] is the originally proposed idea which was improved in Fast simulated annealing (FSA) [50]. FSA converges much faster than CSA while de-trapping from a local minimum more conveniently because of its semi-local visiting distribution. However, for many problems, FSA and GSA cannot find global minima efficiently. In [51], CSA and FSA were generalized according to the Tsallis statistics. This method is known as the dual annealing algorithm. We use *dual annealing algorithm* to solve the optimization model presented in Eq. (28) and obtain an optimal offloading probability (p_i^{VNF}) for each traffic stream i. Dual annealing uses a somewhat distorted Cauchy–Lorentz visiting distribution, whose shape is controlled by parameter q_v,

$$g_{q_v}(\Delta x(t)) \propto \left[T_{q_v}(t)\right]^{-\frac{D}{3-q_v}} \times \left[1+(q_v-1)\frac{[\Delta x(t)]^2}{\left[T_{q_v}(t)\right]^{\frac{2}{3-q_v}}}\right]^{\frac{1}{q_v-1}+\frac{D-1}{2}}, \quad (29)$$

where t is the artificial time and q_v is the visiting parameter that controls the cooling rate, affecting how rapidly the temperature decreases. This visiting distribution is used to generate a trial jump distance $\Delta x(t)$ of variable $x(t)$ under temperature T_{q_v}. D is the dimension of the variable space. The jump is accepted if it is downhill (of the energy function); if it is not, it might be accepted according to an acceptance probability. The artificial temperature T_{q_v} is decreased according to

$$T_{q_v}(t) = T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}. \quad (30)$$

The Metropolis algorithm [52] is used for the acceptance probability, which is calculated as

$$p_{q_a} = \min\{1, \left[1-(1-q_a)\beta\Delta E\right]^{\frac{1}{1-q_a}}\}, \quad (31)$$

where q_a is the acceptance parameter. For $q_a < 1$, zero acceptance probability is assigned to the cases where,

$$[1-(1-q_a)\beta\Delta E] < 0. \quad (32)$$

Table 3
Default values of the simulation and analytical parameters.

Parameter	Default value	Parameter	Default value
λ	55 kilo packets per second (kpps)	λ_i	{9,18,27} kpps
p^{NF}	0.8	p_i^{NF}	{0.8, 0.8, 0.8}
c^{PNF}	$1.2 * 10^9$ million instructions per second (MIPS)	c^{VNF}	$2.4 * 10^9$ MIPS
c^{SP}	150 kpps	c^{SC}	10^9 bits/second
w	20 k MIPS	w_i	{10, 20, 30}k MIPS
l	250 Bytes	l_i	{250, 500, 750} Bytes
D^{SV}	5 µs	N	3

By using the above procedure, the dual annealing algorithm successfully obtains a solution for the optimization model provided in Eq. (28).

5. Evaluation

In this section, we present results obtained from the analytical formulation. We also simulated the network of queues shown in Figs. 1 and 2 and obtained a few simulation results. We decided to validate our analytical model by simulations rather than real measurements because simulation results can be obtained faster while still being able to experiment with different model parameters. For simulations, we built a custom simulator using Simpy (a popular Python Discrete-event simulation library [53]. Fig. 3 shows different components of our simulator and the data flow between them. We use SimPy *processes* and *stores* to implement all the queues. SimPy processes define the simulation behavior by interacting with the SimPy *environment* whenever any event occurs. To model delays at the queues, we use *timeout* events. Upon the occurrence of the timeout event, the process passes a delay value to the environment. In our simulation model, we added labels to the newly generated packets that define the path of the packets in the queuing network. A packet is either labeled as PNF/VNF or NoNF based on the values of p^{NF} and p^{VNF}. The environment time when the packet arrives at the switch is tagged within the packet as metadata. Also, once a packet traverses VNF, it is labeled as "VNF traversed". This tagging aids the switch in differentiating new packets from the packets re-entering the switch post-VNF traversal. Once the packet finishes its execution and reaches the sink (finally comes out of the switch after PNF/VNF processing), the packet's arrival time is subtracted from the current environment time to calculate the delay.

5.1. Parameter settings

The default values used in the simulation and the analytical model for obtaining the numerical results are shown in Table 3. We model three traffic types for heterogeneous traffic cases, each with a different arrival rate (provided in Table 3). The other relevant parameters provided in Table 3 are given as input to the optimization model. The selection of parameters for the simulation and analytical model aligns with the end-to-end latency data provided in [54]. The benchmarks provided in [54] were obtained after experimentation on actual P4 devices.

5.2. Offloading probability vs. average delay

We analyze the impact of VNF offloading on the average packet delay in the first set of results shown in Figs. 4 (for homogeneous case) and 5 (for heterogeneous case). In these two figures, we also compared

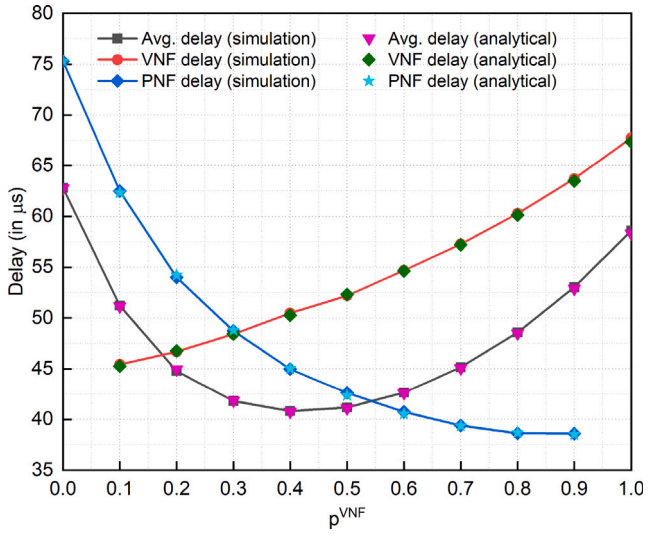


Fig. 4. Comparison of simulation model vs. analytical model for homogeneous traffic.

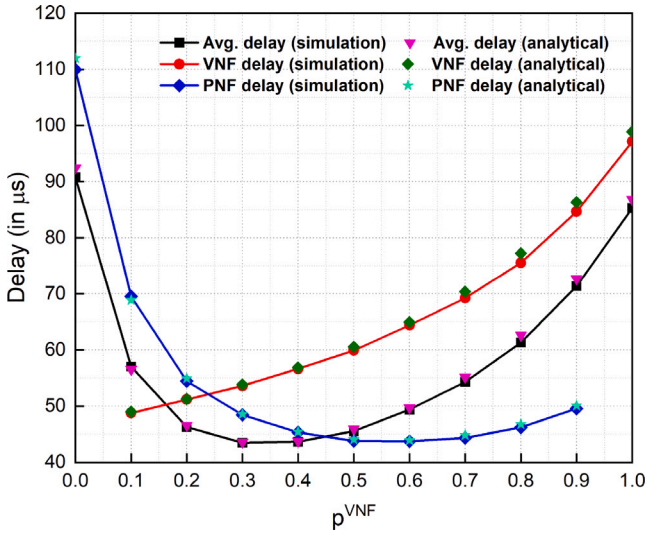


Fig. 5. Comparison of simulation model vs. analytical model for heterogeneous traffic.

the results obtained from the analytical model with the simulation model. We plot the average delay for all the packets and compare it with the two sub-cases of the average delays for the packets traversing VNFs and PNFs. We observe from the figure that the delay values obtained from the analytical and simulation models are almost identical for every value of p^{VNF} for all three cases.

In the average delay plot, initially, when there is no VNF offloading, the PNF queue is highly occupied, resulting in massive delays. As VNF offloading starts, the delay dips as the PNF queue is less congested. However, beyond a point, delay again tends to increase. This is because the impact of the overhead of offloading packets to the VNF is more profound on the delay.

Our second observation from the delay plots is that the PNF delay continuously decreases; however, VNF delay constantly increases with the increase in p^{VNF}. This is because when we increase the offloading probability, the delay for the packets traversing VNF increases as the VNF queue builds up. Similarly, delay for the packets traversing PNF decreases with the rise in offloading probability as the traffic load at the PNF queue decreases.

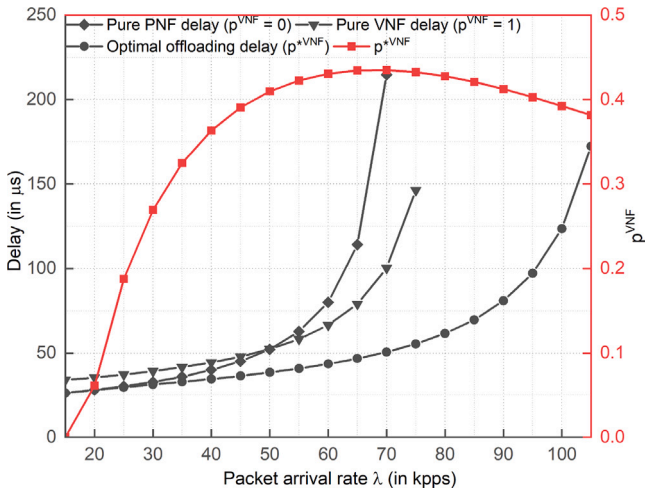


Fig. 6. Impact of packet arrival rate on delay and offloading probability for homogeneous traffic.

Table 4
Flow rates for different cases.

	λ_i		
	f_1	f_2	f_3
Case 1	R	2R	3R
Case 2	3R	R	2R
Case 3	3R	2R	R

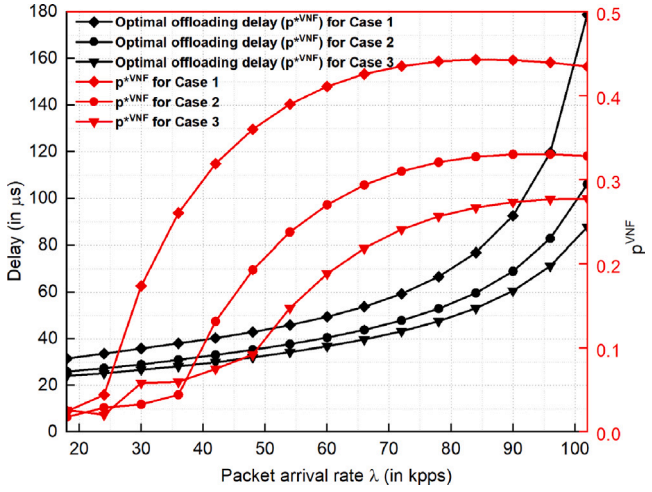


Fig. 7. Impact of packet arrival rate on delay and offloading probability for heterogeneous traffic.

Moreover, for the average delay plot corresponding to the homogeneous traffic case, our algorithm provided the lowest delay of 41.34 µs at the p^{VNF} value of 0.422 (not shown in the figure), which is a correct estimate as per the plot. The takeaway from this result is that it is essential to calculate the optimal offloading probability so that delay can always be minimized.

For the next set of results, we vary different parameters and analyze their impact on the optimal offloading probability (referred to as $p^{*\text{VNF}}$) obtained from the algorithm. To show the performance benefits of our algorithm, for each change in the parameters, we also plot the average packet delay (a) corresponds to $p^{*\text{VNF}}$, denoted as optimal offloading; (b) when there is no offloading, denoted as pure PNF and; (c) when all the packets are offloaded to the VNF, denoted as pure VNF, in the figures.

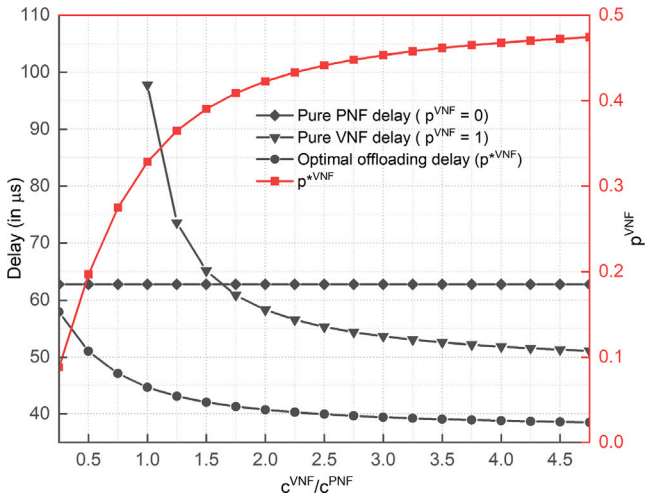


Fig. 8. Impact of VNF and PNF processing capacities on delay and offloading probability for homogeneous traffic.

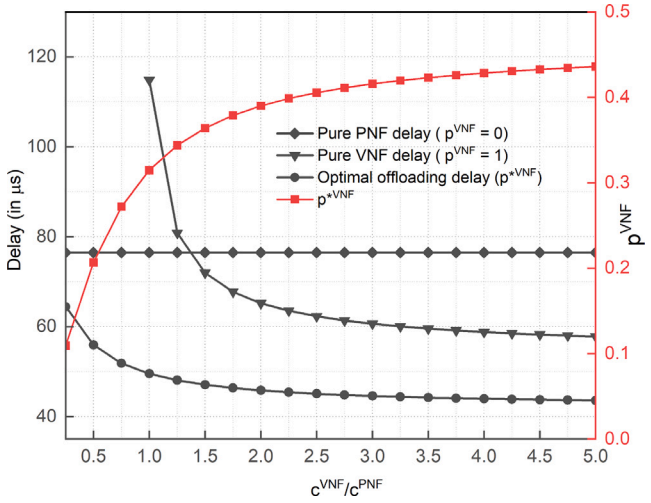


Fig. 9. Impact of VNF and PNF processing capacities on delay and offloading probability for heterogeneous traffic.

5.3. Impact of arrival rate and resource allocation

For the following results shown in Fig. 6, we observe the impact of the arrival rate (λ) on the performance for the homogeneous case. We realize that $p^{*\text{VNF}}$ increases significantly with an initial increase in the arrival rate. However, after an arrival rate of 70 kpps, a further increase results in a drop in the optimal offloading probability. This is because, beyond a particular load, the benefits of offloading start fading because the offloaded packets experience more delay due to queue congestion at the VNF queue and other communication overheads.

The corresponding optimal delay does not increase much with the increase in the packet arrival rate compared to the other two cases of full offloading (the pure VNF case) and no offloading (the pure PNF case). This result is per our expectation because our method estimates the optimal offloading hence the obtained delay is minimum among the three considered situations.

In the case of pure PNF, initially, the delay slightly increases. Then, however, with a further increase in the load, the delay rises abruptly. A similar phenomenon is also observed in the case of pure VNF. For both these cases, the service capacity exhausts after a point. Since VNF capacity is assumed to be twice that of PNF, for VNF, this point comes later than the PNF. The key takeaway from this experiment is that at a

low arrival rate, it does not make sense to offload to the VNF as PNF can handle it without any significant congestion at its queue. However, as the load increases, it is advisable to offload as the queue at the PNF starts building up. Quantitatively, at medium loads (40 k–70 k pps), optimal offloading leads to the performance gains of 13.54%–76.44% and 22.28%–49.55% over a pure PNF and pure VNF solution, respectively.

For the heterogeneous case, we altered the simulation settings to account for the heterogeneity of the traffic. The result shown in Fig. 7 is quite similar to the one for the homogeneous case (i.e., Fig. 6) with a few differences. We have considered three different flows (f_1, f_2, f_3) with different rates R, $2R$ and $3R$ (the mapping of rates to flows is discussed further). The x-axis represents the cumulative packet arrival rate (R+2R+3R). For this result, we analyze three cases with distinct "flow to arrival rate mapping" as shown in Table 4. The values of w_is are 10 k, 20 k and 30 k MIPS and the values of l_is are 250, 500 and 750 Bytes for f_1, f_2 and f_3, respectively. To obtain the plots in Fig. 7, we varied R from 3 k to 17 k pps, with an increment of 1 kpps. For the three cases, the optimal offloading probability ($p_i^{*\text{VNF}}$) and the corresponding average delay values are shown in Fig. 7. All three plots follow a similar pattern for the delay and $p_i^{*\text{VNF}}$. However, in Case 1, the delay is higher because f_3 is mapped to the $3R$ rate. In Case 1, for each x-axis value, 50% of the traffic belongs to f_3, which has the maximum average packet workload (i.e., 30 k MIPS) and the average packet length (i.e., 750 Bytes). For Case 3, the average delay values are the lowest because rate $3R$ is assigned to f_1, which has the lowest average packet workload and length. For an x-axis value of 102 kpps, the delay for Case 1 is 2× that of Case 3. This signifies the impact of the dominant flow's presence (f_3 in this experiment) on average delay. However, this performance difference is not as significant at the lower flow arrival rates. We conclude from this result that in the heterogeneous case, the average delay and optimal offloading probability follow the same trend as in the homogeneous case. However, the curves' slope is heavily dependent upon the average packet workloads and lengths of the dominant flow.

We will answer the next important question regarding how much processing capacity should be allocated to the PNF and the VNF. We observe from Figs. 8 (for homogeneous case) and 9 (for heterogeneous case) that when the VNF's processing capacity is one-fourth to that of the PNF's, the optimal offloading probability is the lowest (< 0.1). However, with an increase in VNF capacity, the optimal offloading probability starts to improve. However, beyond a point, even a sizable increase in the VNF capacity does not yield similar improvements in the optimal offloading probability and the average delay. For the pure VNF case, the delay is exceptionally high at this point, even though the queuing delays at the PNF and the VNF queue are the same. However, other delays, such as propagation delay and additional queuing delays at the switch (due to re-entering of the packet in the switch), add up to the total VNF delay.

The key conclusion from this experiment is that the VNF capacity must be chosen correctly as, beyond a point, we do not get sizable benefits with an increase in the VNF capacity.

5.4. Impact of the propagation delay and requirement of the network function

Next, we examine the impact of the propagation delay between the switch and the VNF in Figs. 10 (homogeneous case) and 11 (heterogeneous case). The results are similar for both of these cases. The optimal offloading probability decreases with an increase in the propagation delay as it directly affects the delay for the packets traveling to and from the VNF. Similarly, the corresponding average delay increases, but not in the same order as the other two cases of pure PNF and VNF. In the pure VNF case, the delay grows linearly with an increase in the switch to VNF propagation delay. However, in the pure PNF case, there is no impact of the propagation delay between the switch to VNF.

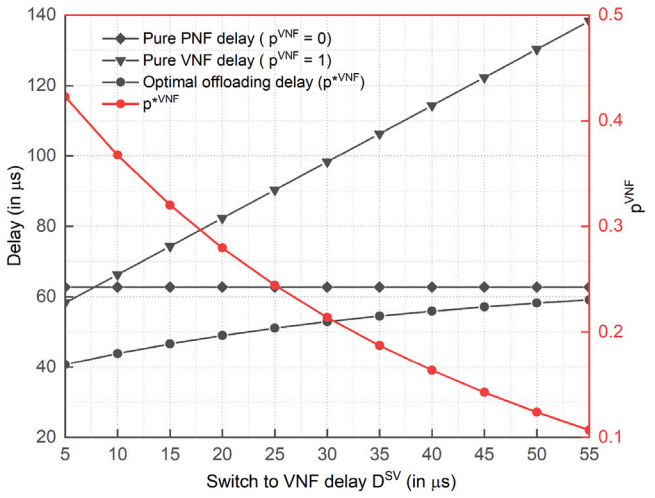


Fig. 10. Impact of propagation delay (between the switch and VNF) on delay and offloading probability for homogeneous traffic.

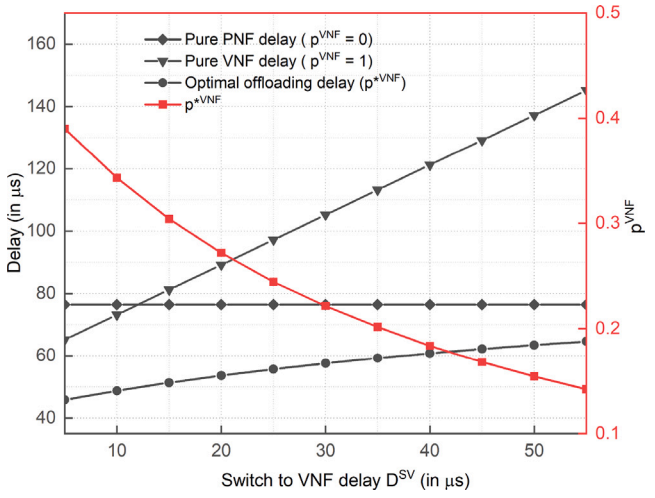


Fig. 11. Impact of propagation delay (between the switch and VNF) on delay and offloading probability for heterogeneous traffic.

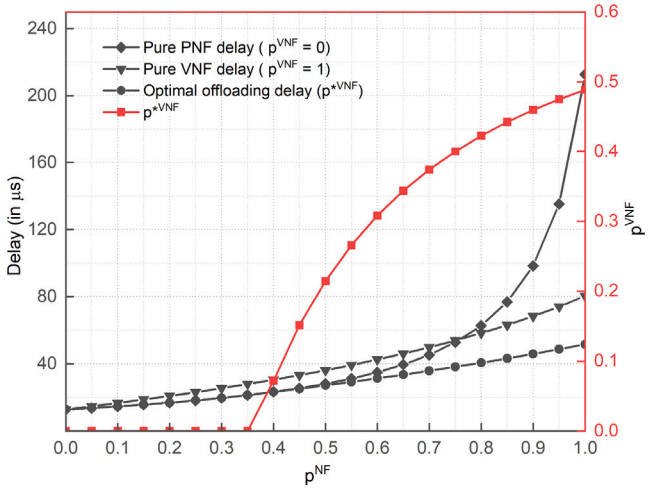


Fig. 12. Impact of network function traversal probability on delay and offloading probability for homogeneous traffic.

Table 5
Probability of network function required for the flows in different cases.

	p^{NF}		
	f_1	f_2	f_3
Case 1	p	$2p$	$3p$
Case 2	$3p$	p	$2p$
Case 3	$3p$	$2p$	p

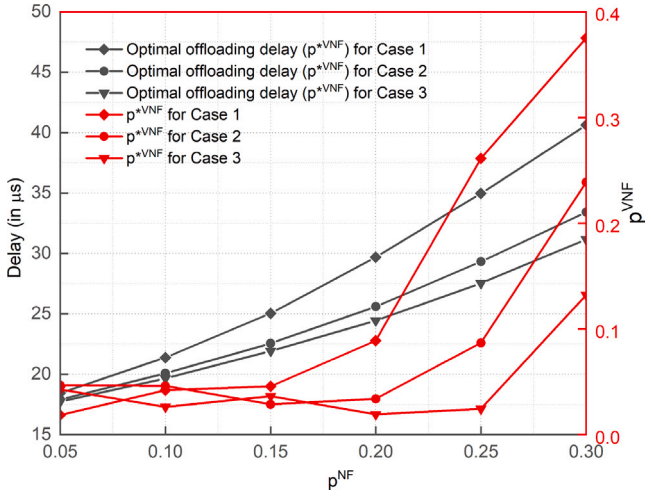


Fig. 13. Impact of network function traversal probability on delay and offloading probability for heterogeneous traffic.

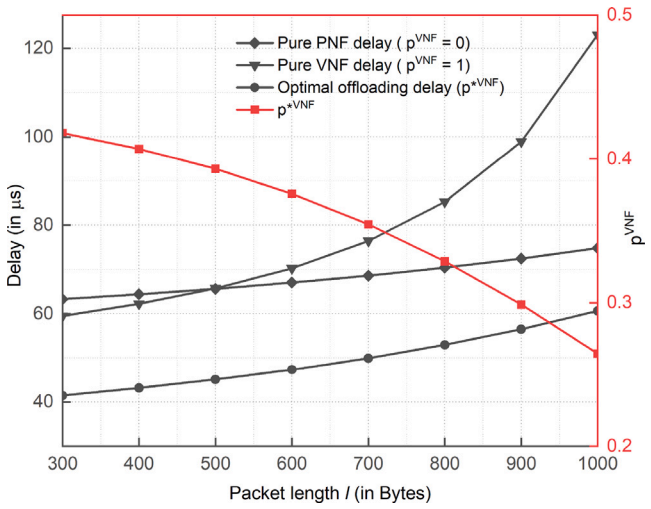


Fig. 14. Impact of packet length on delay and offloading probability for homogeneous traffic.

Table 6
Average packet lengths for the flows in different cases.

	l_i		
	f_1	f_2	f_3
Case 1	L	2L	3L
Case 2	3L	L	2L
Case 3	3L	2L	L

From this experiment, the key message is that the VNF must be placed in close proximity to the programmable switch to obtain the best latency values. This means the VNF must be in a DC that is not too far from the programmable switch; otherwise, the propagation delay will induce less offloading. However, it is interesting to note that with optimal offloading, the overall delay remains under control (60 µs in our case) even with a sizeable one-way propagation delay of 55 µs. In contrast, with pure VNF solution, it proliferates.

We note that not all the packets passing through a programmable switch require traversing the network function. To capture this scenario, we vary the probability of requiring a network function traversal (p^{NF}) in the following experiments and evaluate its impact on the overall performance. For the homogeneous case, we observe from Fig. 12 that the average delay also increases with an increase in p^{NF} for all three delay plots. However, in the case of pure PNF, it grows abruptly beyond the value of 0.6. After this point, the resources at the PNF start exhausting, which results in higher congestion at the PNF queue.

The optimal offloading probability remains zero until the p^{NF} value of 0.4. This is because, up to this point, PNF can efficiently handle all the traffic on its own, so there is no benefit of offloading to the VNF. But after this value, it makes sense to offload to the VNF as the PNF queue starts building. This experiment implies that when fewer packets are required to traverse the network function, offloading does not make sense, as PNF can handle the load itself.

For the heterogeneous traffic scenario, we considered three cases. For these cases, the probability of the network function's requirements for the three flows is listed in Table 5. The result for the heterogeneous case is shown in Fig. 13. In the figure, on the x-axis, we show p values listed in Table 5. It is to be noted that in all three cases, the optimal offloading probability and average delay tend to increase with an increase in p_i^{NF}. For Case 1, the average delay is maximum as f_3, which consists of 50% of traffic (refer Table 3), having the highest probability of packets requiring network function ($3p$). Case 3, having the lowest probability (p) assigned to f_3 incurs the lowest delay among all three cases. It is to be noted that below $p = 0.2$, the optimal offloading probability in all three cases is almost the same, as not much offloading is required for any of the three flows. This is why we observe a random trend for p_i^{*VNF} values correspond to the three cases when p_i^{NF} is less than $p = 0.2$.

5.5. Impact of packet length and workload

Next, in Fig. 14, we show how the packet length impacts the average delay and optimal offloading. As we increase the packet length, the optimal offloading probability decreases. With an increase in the packet size, per-packet transmission time (from the switch to the VNF) increases, resulting in a long queue build-up at the switch's communication queue. With a decrease in the optimal offloading probability, the corresponding average delay obtained from our method increases as the queue at the PNF starts building and the average delay grows.

For the pure VNF case, the impact of packet size on the delay is more profound. In this case, every VNF packet has to be served by the switch's communication queue twice. Additionally, large-sized packets elevate the service time at the communication queue and, eventually, the queue occupancy. Interestingly, delay slightly grows with an increase in packet size, even in the pure PNF case, because packets still have to be handled once by the switch's communication queue.

For the heterogeneous traffic scenario, we have again considered three different flows (f_1, f_2, f_3) with average packet lengths L, $2L$ and $3L$. We show the results for the heterogeneous case in Fig. 15. The x-axis shows L values, which vary from 50 to 850 Bytes, in the increment of 100 Bytes. For this result, "flow to average packet length mapping" is shown in Table 6. The values of λ_is are 9 k, 18 k, and 27 k pps and the values of w_is are 10 k, 20 k and 30 k MIPS for f_1, f_2 and f_3, respectively. For the three cases, the optimal offloading probability (p_i^{*VNF}) and the corresponding average delay values are shown in Fig. 15.

In all three cases, for each x-axis point, 50% of the traffic belongs to f_3 (because f_3 is the dominant flow with λ_i = 27 kpps). For Case 1,

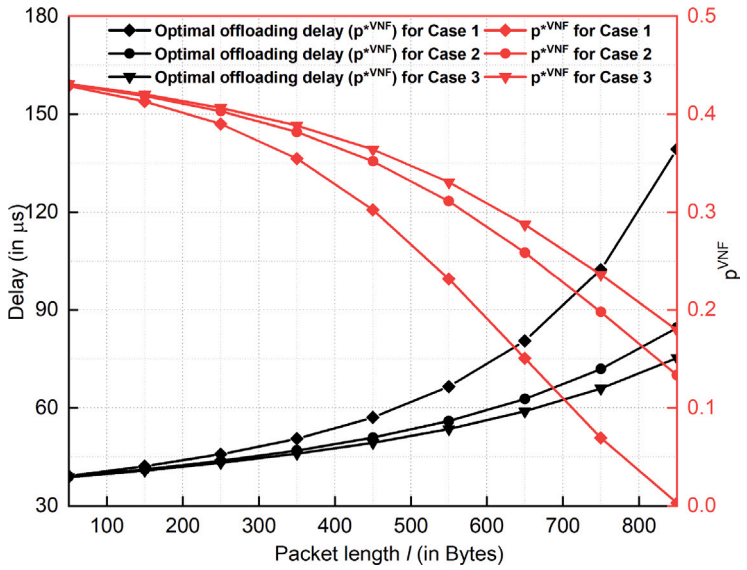


Fig. 15. Impact of packet length on delay and offloading probability for heterogeneous traffic.

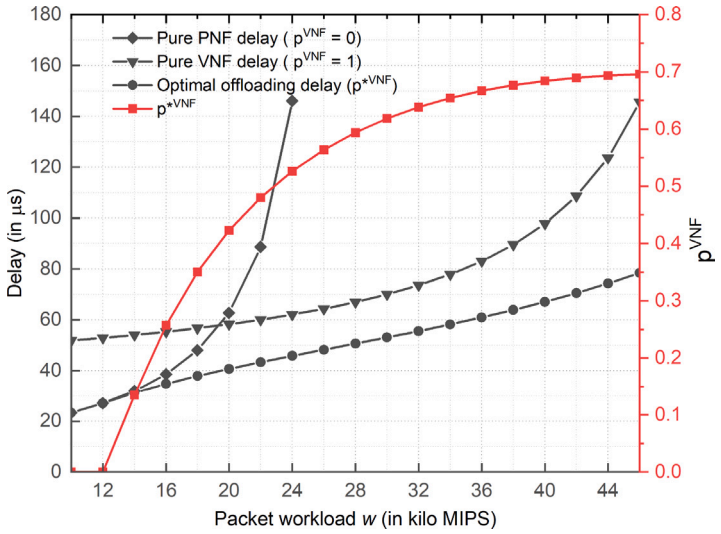


Fig. 16. Impact of packet workload on delay and offloading probability for homogeneous traffic.

the average delay is higher because f_3 is mapped to the average packet size of $3L$, which is the highest among all the cases. This means 50% of the traffic has an average packet size of $3L$, implying more immense propagation delays. For Case 3, the average delay is the lowest among all the cases, as 50% of the traffic has an average packet size of L. These curves indicate the role of the disparate flows' average packet lengths in the aggregate traffic's average delay. For the L value of 850 Bytes, the average delay for Case 1 is 1.85× that of Case 3. This signifies that if a dominant flow has larger packets (2550 Bytes), the delay is exceptionally high compared to the case where the dominant flow has comparably smaller size packets (850 Bytes).

In the following results shown in Fig. 16, we observe the impact of the packet workload on the average delay for the homogeneous case. As we increase the average packet workload, the optimal offloading probability rises. This is because, in our experiments, the VNF processing capacity (c^{VNF}) is assumed to be twice the PNF capacity (c^{PNF}), and it will be utilized fully at a higher packet workload. However, with our method, the delay grows slightly with an increase in the packet workload. The primary reason behind this phenomenon is that even though offloading improves with an increase in packet workload, queue occupancy at the PNF and VNF also increases with the workload.

We show the delay values until the packet workload is less than 25 k MIPS for pure PNF case. The PNF's capacity is exhausted after this workload, so we omit those points from the plot. However, for the pure VNF case, VNF could handle a higher packet workload, but the delay grows significantly after a packet workload of 30 k MIPS. This is because, beyond this workload, the service time at the VNF queue increases remarkably, resulting in higher congestion at the VNF queue.

For the heterogeneous traffic scenario, we have considered three different flows (f_1, f_2, f_3) with average packet workloads of W, $2W$ and $3W$. We show the results for the heterogeneous case in Fig. 17. The x-axis shows W values, varying from 6 k to 26 k MIPS. We again analyze three cases with distinct "flow to average packet workload mapping" as shown in Table 7. The values of λ_is are 9 k, 18 k and 27 k pps and the values of l_is are 250, 500 and 750 Bytes for f_1, f_2 and f_3, respectively. For the three cases, the optimal offloading probability ($p_i^{*\text{VNF}}$) and the corresponding average delay values are shown in Fig. 17.

Similar to the packet length scenario, in all three cases, for each x-axis point, 50% of the traffic belongs to f_3. For Case 1, the average delay is higher because f_3 is mapped to the average packet workload of 3 W, which is the highest among all the cases. This means 50% of

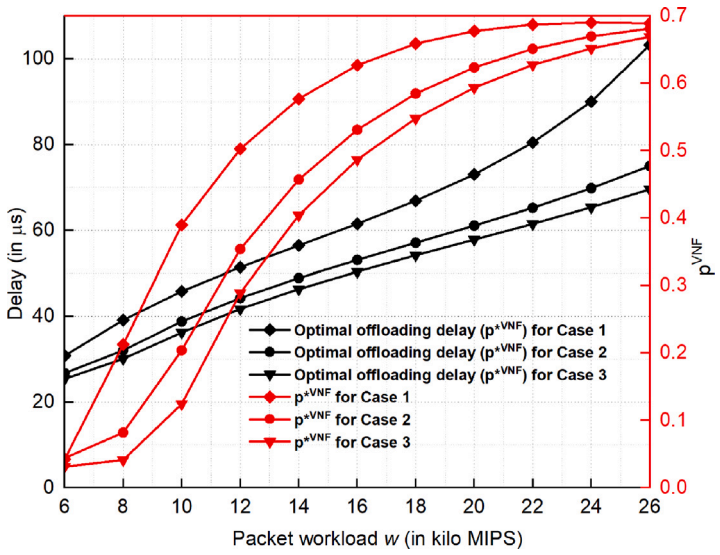


Fig. 17. Impact of packet workload on delay and offloading probability for heterogeneous traffic.

Table 7
Average packet workloads for the flows in different cases.

	w_i		
	f_1	f_2	f_3
Case 1	W	2 W	3 W
Case 2	3 W	W	2 W
Case 3	3 W	2 W	W

the traffic has an average packet workload of 3 W, implying more considerable processing delays (for VNF and PNF). For Case 3, the average delay is the lowest among all the cases, as 50% of the traffic has an average packet workload of W. These results show that the individual flow's workload impacts the aggregate traffic's average packet delay. However, this impact is less pronounced than the impact of packet sizes. This implies that if the flows with different packet workloads are multiplexed, their rates do not heavily impact the delay values.

5.6. Lessons learned for network operators

Based on the results obtained, we have drawn some key conclusions that can benefit an operator planning to offload traffic from a programmable switch to the VNF. Homogeneous case results show significant performance gains of optimal offloading compared to the pure PNF/VNF solutions. Heterogeneous case results are crucial as, most of the time, flows would be multiplexed. A few of these flows would be elephant flows, and others could be mice flows. It is essential to know how the offloading performance changes when the proportion of these flows changes.

Benefits from VNF offloading are highly interlinked with the traffic rate at the P4 switch. For example, VNF offloading makes sense once the traffic load is beyond a threshold. Below that threshold, a PNF or a VNF solitary can provide similar latency values. Once traffic rates cross this threshold, offloading looks promising and the average delay reduces as VNF's capacity salvages the performance loss due to congestion at the PNF queue. After a point, when the arrival rate further increases, offloading benefits are again subsidized as the VNF queues are also congested with traffic. Hence, based on the traffic profile, operators should look for the window where offloading fetches maximum benefits.

Processing capacity of VNF must be in accordance with the PNF capacity at the P4 switch. Since the switch's processing capacity is assumed to be fixed, VNF capacity can be scaled up based on traffic requirements. Our results show that it does not make sense to increase VNF capacity enormously. A VNF capacity of 1.5× the PNF capacity is enough to provide good delay values. Hence operators can plan the capacity of VNF according to what they have in their PNF.

The physical distance between VNF and the switch has a bearing on the overall offloading performance. If the propagation delay between the switch and VNF goes beyond a limit, offloading does not yield any benefits. Hence for operators, it is necessary to place the VNF close to the switch.

Packet attributes also play a crucial role in determining the offloading performance. Since packet size directly affects the transmission delay, the larger the packet size, the more time is required to transmit it to an output port. Offloading from a programmable switch requires transmitting each packet twice from its output ports (once each before and after VNF traversal). This is why flows with large-sized packets are unsuitable for offloading, as the offloading benefit gets neutralized because of the high transmission delay at the switch. On the other hand, packet workload has a reverse relationship with offloading gains. Flows with heavy workloads on packets are better served with offloading. This way, a high-capacity VNF can be utilized fully instead of a PNF hosted in a resource-constrained programmable switch. Hence, operators must note that flows with small-sized packets and heavy workloads are ideal for VNF offloading.

5.7. Model's possible deviations from an actual P4 hardware switch

The P4 switch envisaged in our models (Figs. 1 and 2) is fundamentally similar to an actual P4 hardware switch. However, for modeling purposes, we abstracted specific parameters that may affect delay profiles in an actual P4 switch. Though changes in these parameters could influence the end-to-end delay, precisely quantifying their dynamic impact is analytically challenging. In the next subsections, we describe the actual P4 architecture and our corresponding abstractions.

5.7.1. Portable switch architecture

The switching pipeline on an actual P4 hardware switch follows the Portable Switch Architecture (PSA) [56]. PSA outlines the standard capabilities of a P4 switch that processes and forwards packets through multiple interface ports. PSA defines separate ingress and egress packet pipelines; however, their building blocks are the same. The first component of the PSA ingress/egress pipeline is a parser, which is responsible for header identification, header extraction, buffering, etc. The second component is the control block, which is designed to define the logic of data plane programs, including actions on packets. The control block

Table 8
Measured latency w.r.t. various delay factors in an actual P4 switch [55].

Delay factors in an actual P4 switch while implementing a NF	Measured ingress latency on a P4 switch (@10G Interface)	Measured ingress latency on a P4 switch (@100G Interface)	Corresponding abstraction in this paper
Header extraction (Parsing)	~210-250 ns for 1 to 10 parse states (200-byte packets with 20-byte header)	~155-185 ns for 1 to 10 header states (200-byte packets with 20-byte header)	Switch's Processing Delay
Number of tables in the pipeline	~578-581 ns for 1 to 10 tables (with 3 keys of 32-bit, and 1 k entries in each table)	~276-281 ns for 1 to 10 tables (with 3 keys of 32-bit, and 1 k entries in each table)	Switch's PNF delay
Table key size	~577-581 ns for 8, 16, and 32-bit LPM keys, matched at 5 tables	~278-280 ns for 8, 16, and 32-bit LPM keys, matched at 5 tables	Switch's PNF Delay
Number of table entries	~354-355 ns for 1 k and 100 k entries of 1 key and 32-bit, matched at 5 tables	~225-227 ns for 1 k and 100 k entries of 1 key and 32-bit, matched at 5 tables	Switch's PNF Delay

utilizes stateless objects such as Match-Action tables, as well as stateful external objects like registers. The final component of PSA is a deparser that specifies the packet contents to be sent to the buffer, and what metadata related to the packet is carried with it.

5.7.2. Mapping model's components to the PSA

Table 8 shows results presented in [55] regarding delay profiling of an actual Tofino-based P4 switch (APS BF2556X-1T [57]). The data plane latency provided in [55] is calculated for the traffic running via interfaces configured as 10 Gb (SFP+) and 100 Gb (QSFP28). The table highlights the impact of various attributes on the overall packet delay observed in an actual P4 switch. Though our assumptions are well aligned with the P4 switch's PSA architecture, our focus remained on showing the performance benefits of offloading to an external VNF. Hence, we abstract delays introduced by different building blocks of the P4 switch to processing and PNF delays. In the last column of Table 8, we show our abstractions corresponding to each delay factor. The parsing delay is abstracted to *switch's processing delay*, and the delay induced by the match-action pipeline is considered the *PNF delay*, as this is purely dependent upon how the PNF is implemented on the switch. PNF delay is influenced by several factors, such as the number of match-action tables in the pipeline, table key size, number of entries in the match-action table, and match type (exact or LPM, i.e., longest prefix match). The *switch's communication delay* described in our model is rather straightforward and is directly influenced by the interface speed (10G or 100G).

5.7.3. Impact of PNF type on delay

In addition to the factors mentioned in Table 8, the PNF delay also depends upon the type of network function being implemented on the P4 switch. Because P4 is protocol-independent, vendors can implement packet processing for any data plane protocol. Researchers have already shown that network functions such as L2 forwarding (L2fwd), L3 forwarding (L3fwd), Firewall, Network Address Translation (NAT), Load Balancer, and Intrusion Detection Systems (IDS) can easily be implemented on P4 hardware [12]. All these network functions require a distinct Match+Action pipeline that the P4 program can define. During the compilation, the P4 program can then be converted into a Table Dependency Graph, which is eventually mapped onto the P4 hardware. Note that the network functions mentioned above belong to stateful and stateless categories. For example, L2fwd and L3fwd are stateless, while the firewall is stateful. The abstractions from network functions to P4 pipelines have been studied in [14]. Because of the distinct resource requirements, implementing each PNF on a P4 switch would exhibit a disparate latency profile. Since our proposed analytical models are generic, they do not capture the impact of the type of the PNF being implemented on the P4 switch.

5.7.4. Realization of packet differentiation mechanism on a P4 switch

We show in Figs. 1 and 2 that packets re-enter the P4 switch after traversing VNF. These packets take a different path within the switch compared to those entering it for the first time. Also, for the heterogeneous case, we show that packets from different traffic classes are dealt with differently by the P4 switch. To implement such pipelines, the PSA-based P4 switch supports multiple classes of service for packets sent to the packet buffer. PSA implementations support the class of service mechanism by having a separate FIFO queue per class of service. Thus, unicast packets with the same ingress and egress port but different classes of service may be processed by the egress control block in a different order than the ingress control block processed them. This capability allows the P4 switch to differentiate between packets from various classes, which is crucial for our proposed models.

6. Conclusion

This paper presents the performance benefits of optimally offloading traffic from a P4-switch to a VNF. We have considered two queuing models corresponding to the two cases of homogeneous and heterogeneous traffic streams. We designed a network of M/M/1 queues for the homogeneous case, whereas a network of M/G/1 queues is created to account for the multiplexing of heterogeneous traffic streams. We then utilized queuing formulae to obtain the average packet delay for both cases. Finally, we proposed algorithms to obtain the optimal offloading probability for a packet. The simulation and analytical results demonstrated a significant performance benefit of optimal offloading. The packet offloading probability initially increases with the arrival rate; however, it starts dropping beyond a rate because of the communication–computation trade-off. For homogeneous traffic pattern, with our optimal offloading scheme, at medium loads, performance gains of 13.54%–76.44% and 22.28%–49.55% can be obtained over a pure PNF and pure VNF solution, respectively. The results concluded that offloading probability is maximum for small-sized packets with high workloads. For heterogeneous traffic patterns, the dominant flow affects the overall offloading performance. A delay reduction of up to 2× is obtained if the dominant flow has the smallest average packet size and workload. The results showing the impacts of processing resource allocation and propagation delay (between the P4 switch and VNF) are also presented, which can help an operator plan to deploy P4 switches with the VNFs.

The first area of future work will focus on analyzing the performance of multiple PNFs and VNFs working together to form a service function chain. Supporting multiple PNFs on a P4 switch requires utilizing multiple match-action tables and registers to store state information, which introduces new modeling challenges that need to be addressed. Secondly, we want to consider the MMPP/M/1 model, as next-generation applications such as virtual reality and video streaming generate bursty and correlated traffic. MMPP is an effective tool for

modeling this type of traffic due to its ability to capture time-varying arrival rates. Furthermore, the superposition and splitting operations of MMPPs produce a new MMPP, which facilitates the derivation of analytical models in complex network environments.

CRediT authorship contribution statement

Sidharth Sharma: Conceptualization, Data curation, Investigation, Methodology, Writing – original draft, Writing – review & editing, Software, Visualization. **Yuan-Cheng Lai:** Conceptualization, Methodology, Validation, Visualization, Writing – review & editing. **Ashwin Gumaste:** Resources, Supervision, Writing – review & editing, Validation. **Ying-Dar Lin:** Conceptualization, Project administration, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S. Sharma, A. Kushwaha, M. Alizadeh, G. Varghese, A. Gumaste, Tuneman: Customizing networks to guarantee application bandwidth and latency, ACM Trans. Internet Technol. 23 (1) (2023) 1–26.

[2] R. MacDavid, C. Cascone, P. Lin, B. Padmanabhan, A. Thakur, L. Peterson, J. Rexford, O. Sunay, A P4-based 5G User Plane Function, in: Proceedings of the ACM SIGCOMM Symposium on SDN Research, SOSR, 2021, pp. 162–168.

[3] A. Kushwaha, S. Sharma, N. Bazard, A. Gumaste, Bitstream: A flexible SDN protocol for service provider networks, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–7.

[4] A. Kushwaha, S. Sharma, N. Bazard, T. Das, A. Gumaste, A 400 Gb/s carrier-class SDN white-box design and demonstration: The bitstream approach, J. Lightwave Technol. 36 (15) (2018) 3115–3130.

[5] A. Gumaste, T. Das, S. Sharma, A. Kushwaha, How much NFV should a service provider adopt?, J. Lightwave Technol. 35 (13) (2017) 2598–2611.

[6] A. Gumaste, S. Sharma, T. Das, A. Kushwaha, Analyzing the impact of NFV in large provider networks: A use case perspective, in: 2017 IEEE International Conference on Communications, ICC, IEEE, 2017, pp. 1–7.

[7] A. Kushwaha, S. Sharma, N. Bazard, A. Gumaste, B. Mukherjee, Design, analysis, and a terabit implementation of a source-routing-based SDN data plane, IEEE Syst. J. 15 (1) (2020) 56–67.

[8] J. Woodruff, M. Ramanujam, N. Zilberman, P4dns: In-network dns, in: 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS, IEEE, 2019, pp. 1–6.

[9] R. Kundel, L. Nobach, J. Blendin, H.-J. Kolbe, G. Schyguda, V. Gurevich, B. Koldehofe, R. Steinmetz, P4-BNG: Central office network functions on programmable packet pipelines, in: 2019 15th International Conference on Network and Service Management, CNSM, IEEE, 2019, pp. 1–9.

[10] F. Paolucci, D. Scano, F. Cugini, A. Sgambelluri, L. Valcarenghi, C. Cavazzoni, G. Ferraris, P. Castoldi, User plane function offloading in P4 switches for enhanced 5G mobile edge computing, in: 2021 17th International Conference on the Design of Reliable Communication Networks, DRCN, 2021, pp. 1–3, http://dx.doi.org/10.1109/DRCN51631.2021.9477338.

[11] Intel, Intel® tofino 6.4 tbps, 4 pipelines, 2016, https://www.intel.com/content/www/us/en/products/sku/218643/intel-tofino-6-4-tbps-4-pipelines/specifications.html. (Online; Accessed 20 July 2024).

[12] X. Chen, D. Zhang, X. Wang, K. Zhu, H. Zhou, P4sc: Towards high-performance service function chain implementation on the p4-capable device, in: IFIP/IEEE Symposium on Integrated Network and Service Management, IM, 2019, pp. 1–9.

[13] A. Stockmayer, S. Hinselmann, M. Häberle, M. Menth, Service function chaining based on segment routing using P4 and SR-IOV (P4-SFC), in: International Conference on High Performance Computing, Springer, 2020, pp. 297–309.

[14] M. He, A. Basta, A. Blenk, N. Deric, W. Kellerer, P4nfv: An nfv architecture with flexible data plane reconfiguration, in: 2018 14th International Conference on Network and Service Management, CNSM, IEEE, 2018, pp. 90–98.

[15] C. Sun, J. Bi, Z. Zheng, H. Hu, HYPER: A hybrid high-performance framework for network function virtualization, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2490–2500.

[16] Y. Le, H. Chang, S. Mukherjee, L. Wang, A. Akella, M.M. Swift, T. Lakshman, UNO: Uniflying host and smart NIC offload for flexible packet processing, in: Proceedings of the 2017 Symposium on Cloud Computing, 2017, pp. 506–519.

[17] A. Mohammadkhan, S. Panda, S.G. Kulkarni, K. Ramakrishnan, 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.

[18] T. Osiński, H. Tarasiuk, L. Rajewski, E. Kowalczyk, DPPx: A P4-based data plane programmability and exposure framework to enhance NFV services, in: 2019 IEEE Conference on Network Softwarization (NetSoft), IEEE, 2019, pp. 296–300.

[19] C.-H. Ke, S.-J. Hsu, Load balancing using P4 in software-defined networks, J. Internet Technol. 21 (2020) 1671–1679.

[20] F. Musumeci, A.C. Fidanci, F. Paolucci, F. Cugini, M. Tornatore, Machine-learning-enabled ddos attacks detection in P4 programmable networks, J. Netw. Syst. Manage. 30 (2022) 1–27.

[21] E.F. Kfoury, J. Crichigno, E. Bou-Harb, Offloading media traffic to programmable data plane switches, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–7.

[22] A. Longwell, Barefoot preps for 5G and IoT with its tofino chip and the P4 language, 2018, https://www.sdxcentral.com/articles/news/barefoot-preps-for-5g-and-iot-with-its-tofino-chip-and-the-p4-language/2018/09/. (Online; Accessed 20 July 2024).

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

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

[25] W. Miao, G. Min, Y. Wu, H. Wang, 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, 2015, pp. 60–65.

[26] Z. Shang, K. Wolter, Delay evaluation of openflow network based on queueing model, 2016, arXiv preprint arXiv:1608.06491.

[27] K. Sood, S. Yu, Y. Xiang, Performance analysis of software-defined network switch using $M/Geo/1$ model, IEEE Commun. Lett. 20 (12) (2016) 2522–2525.

[28] W. Miao, G. Min, Y. Wu, H. Wang, J. Hu, Performance modelling and analysis of software-defined networking under bursty multimedia traffic, ACM Tran. Multimedia Comput. Commun. Appl. (TOMM) 12 (5s) (2016) 1–19.

[29] B. Xiong, K. Yang, J. Zhao, W. Li, K. Li, Performance evaluation of OpenFlow-based software-defined networks based on queueing model, Comput. Netw. 102 (2016) 172–185.

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

[31] L.O. Nweke, S.D. Wolthusen, Modelling adversarial flow in software-defined industrial control networks using a queueing network model, in: 2020 IEEE Conference on Communications and Network Security, CNS, 2020, pp. 1–6.

[32] J. Zhao, Z. Hu, B. Xiong, L. Yang, K. Li, Modeling and optimization of packet forwarding performance in software-defined WAN, Future Gener. Comput. Syst. 106 (2020) 412–425.

[33] G. Shen, Q. Li, W. Shi, Y. Jiang, P. Zhang, L. Gu, M. Xu, Modeling and optimization of the data plane in the SDN-based DCN by queuing theory, J. Netw. Comput. Appl. 207 (2022) 103481.

[34] G. Uma Maheswari, K. Vasudevan, Developing a cost model for a multi-controller software-defined network using M/M/c/K queue with retention of reneged flows, in: Proceedings of First International Conference on Computational Electronics for Wireless Communications: ICCWC 2021, Springer, 2022, pp. 271–279.

[35] N. Kröger, F. Mehmeti, H. Harkous, W. Kellerer, Performance analysis of general P4 forwarding devices with controller feedback, in: Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, 2022, pp. 55–64.

[36] Y. Goto, B. Ng, W.K. Seah, Y. Takahashi, Queueing analysis of software defined network with realistic openflow–based switch model, Comput. Netw. 164 (2019) 106892.

[37] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, W.K. Seah, Modelling software-defined networking: Software and hardware switches, J. Netw. Comput. Appl. 122 (2018) 24–36.

[38] Y.-C. Lai, A. Ali, M.S. Hossain, Y.-D. Lin, Performance modeling and analysis of TCP and UDP flows over software defined networks, J. Netw. Comput. Appl. 130 (2019) 76–88.

[39] C. Zhang, H. Yang, G.F. Riley, D.M. Blough, Queueing analysis of auxiliary-connection-enabled switches for software-defined networks, in: 2019 International Conference on Computing, Networking and Communications, ICNC, IEEE, 2019, pp. 497–502.

[40] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, W.K. Seah, Full encapsulation or internal buffering in OpenFlow based hardware switches?, Comput. Netw. 167 (2020) 107033.

[41] W.G. Gadallah, H.M. Ibrahim, N.M. Omar, A seven-dimensional state flow traffic modelling for multi-controller software-defined networks considering multiple switches, Comput. Commun. 196 (2022) 89–108.

[42] K. Mahmood, A. Chilwan, O.N. Østerbø, M. Jarschel, On the modeling of openflow-based sdns: The single node case, 2014, arXiv preprint arXiv:1411.4733.
[43] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, D. Simeonidou, An analytical model for software defined networking: A network calculus-based approach, in: 2013 IEEE Global Communications Conference, GLOBECOM, IEEE, 2013, pp. 1397–1402.
[44] W. Miao, G. Min, Y. Wu, H. Huang, Z. Zhao, H. Wang, C. Luo, Stochastic performance analysis of network function virtualization in future internet, IEEE J. Sel. Areas Commun. 37 (3) (2019) 613–626.
[45] E. Fountoulakis, Q. Liao, N. Pappas, An end-to-end performance analysis for service chaining in a virtualized network, IEEE Open J. Commun. Soc. 1 (2020) 148–163.
[46] Q. Duan, Modeling and performance analysis for service function chaining in the SDN/NFV architecture, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 476–481.
[47] Q. Ye, W. Zhuang, X. Li, J. Rao, End-to-end delay modeling for embedded VNF chains in 5G core networks, IEEE Internet Things J. 6 (1) (2018) 692–704.
[48] J. Billingsley, W. Miao, K. Li, G. Min, N. Georgalas, Performance analysis of SDN and NFV enabled mobile cloud computing, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–6.
[49] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[50] H. Szu, R. Hartley, Fast simulated annealing, Phys. Lett. A 122 (3–4) (1987) 157–162.
[51] Y. Xiang, D. Sun, W. Fan, X. Gong, Generalized simulated annealing algorithm and its application to the thomson model, Phys. Lett. A 233 (3) (1997) 216–220.
[52] G. Bhanot, The metropolis algorithm, Rep. Progr. Phys. 51 (3) (1988) 429.
[53] Team SimPy, SimPy: Discrete event simulation for Python, 2023, https://simpy.readthedocs.io/en/latest/. (Online; Accessed 20 July 2024).
[54] H. Harkous, M. Jarschel, M. He, R. Pries, W. Kellerer, P8: P4 with predictable packet processing performance, IEEE Trans. Netw. Serv. Manag. 18 (3) (2020) 2846–2859.
[55] D. Franco, E.O. Zaballa, M. Zang, A. Atutxa, J. Sasiain, A. Pruski, E. Rojas, M. Higuero, E. Jacob, A comprehensive latency profiling study of the tofino P4 programmable ASIC-based hardware, Comput. Commun. 218 (2024) 14–30.
[56] The P4.org Architecture Working Group, $P4_{16}$ portable switch architecture (PSA), 2021, https://p4.org/p4-spec/docs/PSA.html. (Online; Accessed 20 July 2024).
[57] APS Networks, Advanced programmable switch, 2021, https://www.aps-networks.com/wp-content/uploads/2021/07/210712_APS_BF2556X-1T_V04.pdf. (Online; Accessed 20 July 2024).

Sidharth Sharma received the Ph.D. degree in Computer Science and Engineering from Indian Institute of Technology Bombay, in 2020. Since 2022, he has been an Assistant Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Indore. He is the author of more than 20 research articles and holds two patents. His research interests include Network Softwarization, Network Management, Performance Modeling, and Network Algorithms.

Yuan-Cheng Lai received his Ph.D. degree in Computer Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in 2001 and has been a professor since 2008. His research interests include wireless networks, network performance evaluation, network security, and artificial intelligence.

Ashwin Gumaste received the Ph.D. degree, in 2003. He is an Institute Chair Professor with the Department of Computer Science and Engineering, IIT Bombay, Mumbai. He was the Institute Chair Associate Professor, from 2012 to 2015, and the JR Isaac Chair Assistant Professor, from 2008 to 2011, at IIT Bombay. He has held positions with Fujitsu Laboratories, USA, from 2001 to 2005. He has also worked with Cisco Systems, from 2000 to 2001. He was a consultant to Nokia Siemens Networks, where he focused on the emerging NGPON2. He has also held positions with MIT, from 2008 to 2009, the Lawrence Berkeley National Labs, in 2015, and Iowa State University, in 2009. His work on light trails has been widely referred, deployed, and recognized by both industry and academia. His recent work on omnipresent Ethernet has been adopted by tier-1 service providers and also resulted in the largest-ever acquisition between any IIT and the industry. This has led to a family of transport products under the premise of Carrier Ethernet Switch Routers. His team has built a terabit capable SDN white box for the Ministry of Defense. He has 25 granted U.S. patents and has published about 175 papers in referred conferences and journals. He has authored three books on broadband networks. For research and development contributions, he received the S. S. Bhatnagar Award, in 2018; the DST Swaranajayanti Fellowship, in 2013; the Department of Atomic Energy's SRC Outstanding Research Investigator Award, in 2010; the Department of Space's Vikram Sarabhai Research Award, in 2012; the IBM Faculty Award, in 2012, the National Academy of Sciences (NASI)-Reliance Industries Platinum Jubilee Award, in 2016; the Indian National Academy of Engineering's (INAE) Young Engineer Award, in 2010; and the INAE Fellow, in 2018.

Ying-Dar Lin is a Chair Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010–2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp. He also founded and directed Network Benchmarking Lab (NBL) from 2002, which reviewed network products with real traffic and automated tools, also an approved test lab of the Open Networking Foundation (ONF), and spun off O'Prueba Inc. in 2018. His research interests include machine learning for network security, wireless communications, network softwarization, and mobile edge computing. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), ONF Research Associate (2014–2018), and received K. T. Li Breakthrough Award in 2017 and Research Excellence Award in 2017 and 2020. He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST) with impact factor increased from 9.22 to 25.249 during his term in 2017–2020. He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).