Multi-Timescale Online Optimization of Network Function Virtualization for Service Chaining

Xiaojing Chen, Wei Ni, Tianyi Chen, Iain B. Collings, *Fellow, IEEE*, Xin Wang, *Senior Member, IEEE*, Ren Ping Liu, *Senior Member, IEEE*, and Georgios B. Giannakis, *Fellow, IEEE*

Abstract—Network Function Virtualization (NFV) can cost-efficiently provide network services by running different virtual network functions (VNFs) at different virtual machines (VMs) in a correct order. This can result in strong couplings between the decisions of the VMs on the placement and operations of VNFs. This paper presents a new fully decentralized online approach for optimal placement and operations of VNFs. Building on a new stochastic dual gradient method, our approach decouples the real-time decisions of VMs, asymptotically minimizes the time-average cost of NFV, and stabilizes the backlogs of network services with a cost-backlog tradeoff of $[\epsilon, 1/\epsilon]$, for any $\epsilon > 0$. Our approach can be relaxed into multiple timescales to have VNFs (re)placed at a larger timescale and hence alleviate service interruptions. While proved to preserve the asymptotic optimality, the larger timescale can slow down the optimal placement of VNFs. A learn-and-adapt strategy is further designed to speed the placement up with an improved tradeoff $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$. Numerical results show that the proposed method is able to reduce the time-average cost of NFV by 23% and reduce the queue length (or delay) by 74%, as compared to existing benchmarks.

Index Terms—Network Function Virtualization, virtual machine, distributed optimization, stochastic approximation.

1 Introduction

Decoupling dedicated hardware from network services and replacing them with programmable virtual machines (VMs), Network Function Virtualization (NFV) is able to provide critical network functions on top of optimally shared physical infrastructure [1], [2]. This can avoid disproportional hardware investments on short-lived functions, and adapt quickly as network functions evolve [3]. Particularly, a virtual network function (VNF) is a virtualized task formerly carried out by proprietary and dedicated hardware, which moves network functions out of dedicated hardware devices and into software [3].

A network service (or service chain) can consist of multiple VNFs which need to be run in a predefined order at different VMs running different VNF instances (i.e., software) [4]–[6]. Challenges arise from making optimal online decisions on the placement of VNFs, and the processing and routing of network services at each VM, especially in large-scale network platforms. On one hand, given the sequence of VNFs to be executed per network service, the optimal decisions of individual VMs are coupled. On the other hand, stochasticity prevails in the arrivals of network services, and the link capacity between VMs stemming from concurrent traffic [7]. Prices can also vary for the service of a VM, depending on the pricing policy of the service providers. The possibility of leveraging temporal resource variability implies the couplings of optimal decisions over time [8], [9]. Other challenges also include limited scalability resulting from centralized designs [10].

These are open problems and have not been captured in previous works on VNF placement. The work in [11] focused on the placement of VNFs under the assumption of persistent arrivals of network services, where network services were instantly processed at the VMs admitting them and network service chains cannot be supported. The work in [12] addressed the placement of VNFs in a capacitated cloud network. The placement problem was formulated as a generalization of the Facility Location and Generalized Assignment problem; near-optimal solutions were provided with bi-criteria constant approximations. However, the model in [12] cannot account for function ordering or flow routing optimization.

Taking network service chains into account, recent works studied optimal decision-makings on processing and routing network services, under the assumption of the availability of the a-priori knowledge on service arrivals [13]. NP-complete mixed integer linear programming (MILP) was formulated to minimize the delay of network service chains [10]. A heuristic genetic approach was developed to

Work in this paper was supported by the National Natural Science Foundation of China grant 61671154, the National Key Research and Development Program of China grant 2017YFB0403402, and the Innovation Program of Shanghai Municipal Science and Technology Commission grant 17510710400; US NSF grants 1509005, 1508993, 1423316, and 1442686.

- *X. Chen is with the Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai, China. E-mail: jodiechen@shu.edu.cn. She was with Fudan University, Shanghai, China, and Macquarie University, Sydney, Australia.*
- *W. Ni is with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, NSW 2122, Australia. E-mail: wei.ni@data61.csiro.au.*
- *T. Chen and G. B. Giannakis are with the Dept. of Electrical and Computer Engineering and the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455 USA. Emails: {chen3827, georgios}@umn.edu.*
- *I. B. Collings is with the School of Engineering, Macquarie University, Sydney, NSW 2109, Australia. Email: iain.collings@mq.edu.au.*
- *X. Wang is with the Shanghai Institute for Advanced Communication and Data Science, Key Laboratory for Information Science of Electromagnetic Waves (MoE), the Dept. of Communication Science and Engineering, Fudan University, 220 Handan Road, Shanghai, China. Email: xwang11@fudan.edu.cn.*
- *R. P. Liu is with the School of Electrical and Data Engineering, University of Technology Sydney, Sydney, NSW 2007, Australia. Email: RenPing.Liu@uts.edu.au.*

solve MILP by sacrificing optimality [10]. Greedy algorithms were developed to minimize flowtime or cost, or maximize revenue at a snapshot of the network [14]. These heuristic methods need to run in a centralized manner, thereby limiting scalability. Moreover, none of them have taken random service arrivals or dynamic pricing into account.

In this paper, we propose a new approach to distributed online optimization of NFV, where asymptotically optimal decisions on the placement and operation of NFV are adaptively generated at individual VMs. The proposed approach is able to significantly enhance the scalability of NFV for large-scale computing platforms, such as the emerging mobile edge computing (MEC) and pervasive computing. Accounting for random service arrivals and time-varying prices, a stochastic dual gradient method is developed to decouple optimal decision-makings across different VMs and different time slots. It minimizes the time-average cost of NFV while stabilizing the queues of VMs. The gradients can be interpreted as the backlogs of the queues at every VM, and updated locally by the VM. With a proved cost-delay tradeoff $[\epsilon, 1/\epsilon]$, the proposed method is able to asymptotically approach the global optimum which would be generated offline at a prohibitive complexity, by tuning the coefficient ϵ.

Another key contribution of the paper is that we extend the distributed online optimization of NFV to two timescales, where the placement of VNFs is carried out at the VMs at a much larger time interval than the operations of the VNFs. This can effectively alleviate the interruptions that the placement/installation of VNFs can cause to network service provisions. We prove that the optimality loss of the two-timescale placement and operation of NFV is upper bounded, and the asymptotic optimality of the proposed distributed online optimization is preserved. Other contribution is that we further speed up the placement of VNFs and improve the cost-delay tradeoff to $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$ by designing a learn-and-adapt approach, where the statistics of network dynamics is learned from history with increasing accuracy.

Corroborated by numerical results, the proposed approach is able to reduce the time-average cost of NFV by 23% and reduce the queue lengths (or delays) by 74%, as compared to existing non-stochastic approaches. Accounting for service chaining in stochastic NFV scenarios, the proposed algorithm is important and practical. NFV decouples functions, services and software from dedicated hardware, and changes the architecture of the networks [15]. The proposed approach further decouples decisions and operations of NFV between network entities (e.g., VMs) to enhance the scalability of NFV, and hence has the potential to modernize the architecture of mobile networks.

In a different yet relevant context, stochastic optimization has been developed for single- or multi-timescale resource allocation [16]–[22], routing [23], [24], and service computing [25] in queueing systems to deal with stochastic arrivals of workloads or energy. In particular, backpressure routing algorithms were developed in [23], [24] to maximize throughput or minimize time-average costs in distributed (wireless) mesh networks, while stabilizing the queues across the networks. Yet, the backpressure algorithms were only focused on the routing of workloads, and did not involve any workload processing. An energy-efficient offloading algorithm was proposed for mobile computing in [25], which can dynamically offload part of an application's computation request to a dedicated server. None of the existing approaches [16]–[25] have taken sequentially chained network services into account. Distinctively different from the existing methods, our approach is able to process and offload/forward sequentially chained network services which strongly couple the optimal decisions of VMs on routing and processing in time and space (i.e., among VMs). In particular, our approach decouples the strong coupling, optimizes both the processing and offloading of chained network services, and substantially reduces the queue lengths (or delays) by taking a learn-and-adapt strategy.

While part of the content has been presented in our conference version [26], this paper has a number of new and important contributions. Specifically, this paper optimizes the online placement of VNFs, apart from the operations of NFV such as transmission and processing of tasks; and proves the asymptotic optimality of the placement and operations. The paper generalizes the online optimization of the VNF placement and operations to multiple different timescales, so that the disruption/interruption of operations caused by the replacement of VNFs can be effectively alleviated without compromising the asymptotic optimality. Moreover, the paper provides full details on derivation, optimization and optimality proof. In contrast, our earlier work [26] only optimized the operations at a single timescale, given the placement of VNFs (or in other words, VNFs were deployed in prior). In addition, detailed proofs and analyses were not provided in [26].

The rest of this paper is organized as follows. In Section II, the system model is described. In Section III, the distributed online optimization of the placement of VNFs and the processing/routing of network services is developed. Optimal placement and operation of NFV are investigated at two different timescales in Section IV to prevent its interruptions to network service provisions. Numerical tests are provided in Section V, followed by concluding remarks in Section VI. Notations in the paper are listed in Table I.

2 System Model

Consider a platform consisting of N VMs, supporting K VNFs, and operating in a (possibly infinite) scheduling horizon consisting of T slots. The slotted design is feasible, as Network Timing Protocol (NTP) and its variations have been extensively adopted in networks and neighboring devices/servers can be synchronized with a typical offset of microseconds [27]. The slotted design is also efficient to exchange information and coordinate operations among neighboring VMs (on a slot basis). A VM can efficiently utilize every slot to either transmit or receive tasks, and process tasks, hence making the effective use of its virtual links and processor.

The VMs can be located separately at different host servers, or co-located at the same host server. Assume that every VM can admit network services, and output the results; see Fig. 1. This is consistent with existing designs of NFV systems. Let $\mathcal{N} = \{1, \ldots, N\}$ collect the N VMs. Let

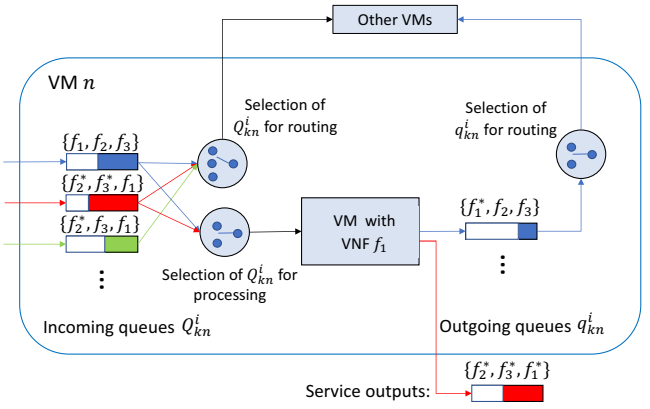


Fig. 1. An illustration on the processing and routing procedure of network services within VM n, where f^* indicates processed VNFs.

f_k $(k = 1, \ldots, K)$ denote the k-th VNF which can only be processed at the VMs running the corresponding software.

Assume that every VM runs a single VNF. Let $e_{kn}(t) \in \{0, 1\}, \forall k, n, t$, denote the ability of VM n to process f_k at time t. Let $e_{kn}(t) = 1$, if VM n is installed with VNF f_k; and $e_{kn}(t) = 0$, otherwise. We have $\sum_k e_{kn}(t) = 1, \forall n, t$, which specifies that every VM runs a single VNF. This is consistent with typical configurations of VMs [28]. The proposed algorithm can be readily extended to general scenarios where a VM runs multiple VNFs, as will be discussed in Section 3.2.

Let $\mathcal{I}$ collect all possible types of network services, and each type of network service is to be processed by a permuted sequence of $\{f_1, \ldots, f_K\}$ or its subset. A network service needs to traverse among multiple VMs, until the service are processed by the related VNFs in correct order. We design up to $2|\mathcal{I}|K$ service queues at each VM n. (For analytic tractability, we assume here that the VMs have sufficient memories, and the queues do not overflow. Nevertheless, one of the constraints we consider in this paper is that the time-average lengths of the queues are finite. As will be proved in Section III-A, the time-average lengths of the queues can be adjusted and reduced through parameter reconfiguration.) Half of the queues buffers network services to be processed by f_k $(k = 1, \ldots, K)$; and the other half buffers the results of the first half after processed by f_k and to be routed to downstream VMs for further processing.

Let $Q^i_{kn}(t)$ denote the queue lengths of type-i network services to be processed by VNF f_k at VM n per slot t. Let $q^i_{kn}(t)$ denote the queue lengths of type-i network services after processed by VNF $f_{k'}$ at VM n, and to be processed by VNF f_k at downstream VMs per slot t. Here, $f_{k'}$ denotes the VNF which needs to be run prior to f_k for type-i network services; $\mathbf{Q}(t) = \{Q^i_{kn}(t), \forall n \in \mathcal{N}, i \in \mathcal{I}, k = 1, \ldots, K\}$, $\mathbf{q}(t) = \{q^i_{kn}(t), \forall n \in \mathcal{N}, i \in \mathcal{I}, k = 1, \ldots, K\}$, and $\mathbf{A}(t) = \{\mathbf{Q}(t), \mathbf{q}(t)\}$. Let $R^{i,t}_{kn} \leq R^{\max}$ denote the arrival rate (in KB/slot) of new type-i network services to be processed by VNF f_k at VM n, where $R^{\max}$ is the maximum arrival rate.

Note that, placing all network services in one queue, and placing the network services in a large number of queues based on the types of services, are just different ways of arranging the storage at a VM, and would not markedly change the requirement of the total storage memory of the VM at all. This is due to the fact that the memory requirement of the VM only depends on the total amount of the services. The storage memory of the server would not expand, as the number of VNF permutations increases while the total amount of services does not.

Let $u^i_{k,ab}(t)$ denote the transmit rate (in KB/slot) for transmitting, over link $[a, b]$ (i.e., from VM a to VM b) at time slot t, the type-i services to be run by VNF f_k. Let $v^i_{k,ab}(t)$ denote the transmit rate for transmitting, over link $[a, b]$ (i.e., from VM a to VM b) at time slot t, the type-i services which have been run at VM a and are to be further run by VNF f_k

TABLE 1
Notation and Definition

Notation	*Definition*
N, K, $\lvert\mathcal{I}\rvert$	Number of VMs, VNFs and types of network services, respectively
T	Total number of scheduling time slots
f_k	The kth VNF
Q^i_{kn}	Queue length of type-i network services to be processed by VNF f_k at VM n
q^i_{kn}	Queue length of type-i network services after processed at VM n and to be further processed by VNF f_k at downstream VMs
$\mathbf{A}(t)$	$\mathbf{A}(t) = \{\mathbf{Q}(t), \mathbf{q}(t)\}$
$R^{i,t}_{kn}$	Arrival rate of new type-i network services to be processed by VNF f_k at VM n
α^t_n	Time-varying price for service processing at VM n
$\beta^t_{[a,b]}$	Time-varying price for service routing over link $[a, b]$
$R^{\max}$, $l^{\max}_{ab}$, $p^{\max}_n$	The maximum arrival rate, the maximum transmit rate of link $[a, b]$, and the maximum processing rate of VM n, respectively
Φ^t	The total cost of routing services over all links and running VNFs on all VMs per slot t
$\mathbf{x}^t$, $\mathcal{X}$	$\mathbf{x}^t := \{\mathbf{e}^t, \mathbf{p}^t, \mathbf{u}^t, \mathbf{v}^t\}$ and $\mathcal{X} := \{\mathbf{x}^t, \forall t\}$
$\boldsymbol{s}^t$	$\boldsymbol{s}^t := [R^{i,t}_{kn}, \alpha^t_n, \beta^t_{[a,b]}, \forall [a, b], i, k, n]$
$\lambda^i_{kn,1}$, $\lambda^i_{kn,2}$	Lagrange multipliers
$g_{\lambda^i_{kn,1}}$, $g_{\lambda^i_{kn,2}}$	Gradient
T_Δ	Number of time slots within a large time interval
e_{kn}	Binary decision to be optimized for installing f_k at VM n
$u^i_{k,ab}$	Transmit rate to be optimized for transmitting, over link $[a, b]$ (i.e., from VM a to VM b) at time slot t, the type-i services to be run by VNF f_k
$v^i_{k,ab}$	Transmit rate to be optimized for transmitting, over link $[a, b]$ (i.e., from VM a to VM b) at time slot t, the type-i services which have been run at VM a and are to be further run by VNF f_k at other VMs
p^i_{kn}	Processing rate of VM n to be optimized for type-i network services to be processed by VNF f_k

at other VMs. It is easy to see that at any time t, we have

$$u^i_{k,ab}(t) \geq 0, \;\; v^i_{k,ab}(t) \geq 0, \;\; \forall i,k,[a,b], \\ \sum_{i,k}[u^i_{k,ab}(t) + v^i_{k,ab}(t)] \leq l^{\max}_{ab}, \tag{1}$$

where $l^{\max}_{ab}$ is the maximum transmit rate of link $[a,b]$, representing the capacity constraint of the link. The maximum transmit rates (or in other words, the link bandwidths) are specified for the links. The instantaneous transmit rates of the links are lower than the maximum transmit rates, and can be random and time-varying due to random background traffic.

Let $p^i_{kn}(t)$ denote the processing rate of VM n (in KB/slot) for the type-i network services to be processed by VNF f_k at time slot t. We have

$$p^i_{kn}(t) \geq 0, \;\; \sum_{i,k} p^i_{kn}(t) e_{kn}(t) \leq p^{\max}_n, \; \forall i,k,n, \tag{2}$$

where $p^{\max}_n$ is the maximum processing rate of VM n, representing the capacity constraint of the VM. The VMs are preconfigured. Our decisions are on which VNF to place on each VM, and the schedule of the VMs running the VNFs. The preconfiguration of the VMs decouples the physical resources and software, and facilitates NFV. The preconfigured VMs (or the underlaying physical resources) can be efficiently utilized by jointly optimizing the placement and schedule of the VNFs executed on the VMs. This can dramatically simplify the network initialization which can be ineffective and quickly outdated anyway due to the time-varying traffic arrivals and network conditions. Moreover, the preconfiguration of VMs is particularly reasonable in distributed platforms, such as MEC, where VMs can run at distributed network nodes (e.g., server and switch). Each VM counts on the physical resources of an individual network node.

Therefore, the queue length of type-i network services to be processed by VNF f_k at VM n follows, $\forall i,k,n,t$,

$$Q^i_{kn}(t+1) = \max\{Q^i_{kn}(t) - \sum_{b\in\mathcal{N}} u^i_{k,nb}(t) - p^i_{kn}(t)e_{kn}(t), 0\} \\ + \sum_{a\in\mathcal{N}} u^i_{k,an}(t) + \sum_{c\in\mathcal{N}} v^i_{k,cn}(t) + R^{i,t}_{kn}. \tag{3}$$

The queue length of type-i network services after processed by VNF $f_{k'}$ at VM n, and to be processed by VNF f_k at downstream VMs follows, $\forall i,k,n,t$,

$$q^i_{kn}(t+1) = \max\{q^i_{kn}(t) - \sum_{d\in\mathcal{N}} v^i_{k,nd}(t), 0\} + p^i_{k'n}(t)e_{k'n}(t), \tag{4}$$

where $q^i_{kn}(t) = 0$ ($k \in \emptyset$) in the case that the type-i network services complete processing at the terminal VM n and are output from the platform.

We define the network is *stable* if and only if the following is met [29] (recall that $\mathbf{A}(t) = \{\mathbf{Q}(t), \mathbf{q}(t)\}$):

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[|\mathbf{A}(t)|] \leq \infty. \tag{5}$$

Considering the processing and routing cost of network services in the platform, we define the total cost of routing services over all links and running VNFs on all VMs per slot t, as given by:

$$\Phi^t(\mathbf{e}^t, \mathbf{p}^t, \mathbf{u}^t, \mathbf{v}^t) := \phi^t_1(\mathbf{e}^t, \mathbf{p}^t) + \phi^t_2(\mathbf{u}^t, \mathbf{v}^t), \tag{6}$$

where $\mathbf{e}^t = \{e_{kn}(t), \forall k,n\}$, $\mathbf{p}^t = \{p^i_{kn}(t), \forall i,k,n\}$, $\mathbf{u}^t = \{u^i_{k,ab}(t), \forall i,k,[a,b]\}$, and $\mathbf{v}^t = \{v^i_{k,ab}(t), \forall i,k,[a,b]\}$. In (6), $\phi^t_1(\mathbf{e}^t, \mathbf{p}^t) = \sum_{i,k,n} \alpha^t_n (p^i_{kn}(t)e_{kn}(t))^2$ and $\phi^t_2(\mathbf{u}^t, \mathbf{v}^t) = \sum_{a,b,i,k} \beta^t_{[a,b]}[(u^i_{k,ab}(t))^2 + (v^i_{k,ab}(t))^2]$ are the costs that the network service provider charges for usages of VMs and links, respectively, e.g., following a quadratic pricing policy [30], [31]. For non-elastic network services, e.g., video streaming and multimedia applications, these with urgent demand for high resources (i.e., bandwidths and CPUs) would charge for high prices [30]. Here, α^t_n is the time-varying price for service processing at VM n and $\beta^t_{[a,b]}$ is the time-varying price for service routing over link $[a,b]$.

3 Distributed Online Optimization of Placement and Operation of NFV

In this section, the optimization problem of the placement and operation of NFV is to be established, where the input of the problem is the current time-varying prices for service processing and routing variables $\{\alpha^t_n, \beta^t_{[a,b]}, \forall [a,b], n, t\}$, and the random service arrivals $\{R^{i,t}_{kn}, \forall i,k,n,t\}$. The objective of the problem is to minimize the time-average cost of NFV on a network platform while preserving the stability of the platform [cf. (5)], under random network service arrivals and prices. This can be achieved by making stochastically optimal decisions on processing or routing network services at every VM and time slot in a distributed fashion. Let $\mathbf{x}^t := \{\mathbf{e}^t, \mathbf{p}^t, \mathbf{u}^t, \mathbf{v}^t\}$ and $\mathcal{X} := \{\mathbf{x}^t, \forall t\}$. The problem of interest is to solve

$$\Phi^* = \min_{\mathcal{X}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{\Phi^t(\mathbf{x}^t)\} \\ \text{s.t.} \quad (1),\ (2),\ (3),\ (4),\ (5),\ \forall t \tag{7}$$

where the expectation of $\Phi^t(\mathbf{x}^t)$ is taken over all randomnesses. The service arrival rate $\{R^{i,t}_{kn}, \forall i,k,n,t\}$ and the routing and processing prices $\{\alpha^t_n, \beta^t_{[a,b]}, \forall [a,b], n, t\}$ are all random.

We assume here an ideal network setting where any VM may execute different VNF at different time instants, and derive the asymptotically optimal solution under the ideal setting. The solution provides the stepping stone to the solution for the more realistic setting where changes of VNF incur costs (i.e., disruptions/interruptions to NFV operations) and therefore can only take place once for a while at a different timescale from the execution of VNF, as will be described in Section 4.

3.1 Dual gradient and asymptotic optimality

It is difficult to solve (7) since we aim to minimize the average cost over an infinite time horizon. In particular, the queue dynamics in (3) and (4) couple the optimization variables in time, rendering intractability for traditional solvers. Combining (3) and (4) with (5), however, it can be shown that in the long term, the service processing and routing

rates must satisfy the following necessary conditions of queue stability [29]

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\sum_{a\in\mathcal{N}}u^i_{k,an}(t)+\sum_{c\in\mathcal{N}}v^i_{k,cn}(t)+R^{i,t}_{kn} - \sum_{b\in\mathcal{N}}u^i_{k,nb}(t)-p^i_{kn}(t)e_{kn}(t)]\le 0,\ \forall i,k,n. \tag{8a}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[p^i_{k'n}(t)e_{k'n}(t)-\sum_{d\in\mathcal{N}}v^i_{k,nd}(t)]\le 0,\forall i,k,k',n. \tag{8b}$$

As a result, (7) can be relaxed as

$$\tilde{\Phi}^*=\min_{\mathcal{X}}\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\{\Phi^t(\mathbf{x}^t)\},\quad \text{s.t.}\quad (1),\ (2),\ (8),\ \forall t. \tag{9}$$

Compared to (7), problem (9) eliminates the time coupling among variables $\{\mathbf{A}(t),\forall t\}$ by replacing (3), (4) and (5) with (8). Since (9) is a relaxation of (7) with its optimal objective $\tilde{\Phi}^*\le\Phi^*$, if one solves (9) instead of (7), it is prudent to derive the optimality bound of Φ^*, provided that the solution $\mathcal{X}$ for (9) is feasible for (3), (4) and (5), as will be shown in Theorem 1.

We can take a stochastic gradient approach to solving (9) with asymptotical optimality guarantee. Concatenate the random parameters into a state vector $\boldsymbol{s}^t:=[R^{i,t}_{kn},\alpha^t_n,\beta^t_{[a,b]},\forall[a,b],i,k,n]$. For analytic tractability, $\boldsymbol{s}^t$ is assumed to be independent and identically distributed (i.i.d.) across slots. (In practice, $\boldsymbol{s}^t$ can be non-i.i.d. and even correlated. In such case, the algorithm proposed in this paper can be readily applied. Yet, performance analyses of the non-i.i.d. case can be obtained by generalizing the delayed Lyapunov drift techniques [29]). Then, we can rewrite (9) as

$$\tilde{\Phi}^*=\min_{\mathcal{X}}\quad \mathbb{E}\{\Phi^t(\mathcal{X}(\boldsymbol{s}^t);\boldsymbol{s}^t)\} \tag{10a}$$

$$\text{s.t.}\quad u^i_{k,ab}(\boldsymbol{s}^t)\ge 0,$$

$$\sum_{i,k}[u^i_{k,ab}(\boldsymbol{s}^t)+v^i_{k,ab}(\boldsymbol{s}^t)]=u^{i^*}_{k^*,ab}(\boldsymbol{s}^t)(\text{or } v^{i^*}_{k^*,ab}(\boldsymbol{s}^t))\le l^{\max}_{ab}, \tag{10b}$$

$$p^i_{kn}(\boldsymbol{s}^t)\ge 0,\ \sum_{i,k}p^i_{kn}(\boldsymbol{s}^t)e_{kn}(\boldsymbol{s}^t)=p^{i^*}_{k^*n}(\boldsymbol{s}^t)\le p^{\max}_n, \tag{10c}$$

$$\mathbb{E}[\sum_{a\in\mathcal{N}}u^i_{k,an}(\boldsymbol{s}^t)+\sum_{c\in\mathcal{N}}v^i_{k,cn}(\boldsymbol{s}^t)+R^{i,t}_{kn} - \sum_{b\in\mathcal{N}}u^i_{k,nb}(\boldsymbol{s}^t)-p^i_{kn}(\boldsymbol{s}^t)e_{kn}(\boldsymbol{s}^t)]\le 0, \tag{10d}$$

$$\mathbb{E}[p^i_{k'n}(\boldsymbol{s}^t)e_{k'n}(\boldsymbol{s}^t)-\sum_{d\in\mathcal{N}}v^i_{k,nd}(\boldsymbol{s}^t)]\le 0, \tag{10e}$$

where $p^i_{kn}(\boldsymbol{s}^t):=p^i_{kn}(t), e_{kn}(\boldsymbol{s}^t)=e_{kn}(t), u^i_{k,ab}(\boldsymbol{s}^t):=u^i_{k,ab}(t), v^i_{k,ab}(\boldsymbol{s}^t):=v^i_{k,ab}(t),\ \forall[a,b],i,k,n$, and $\Phi^t(\mathcal{X}(\boldsymbol{s}^t);\boldsymbol{s}^t):=\Phi^t(\mathbf{x}^t)$. Formulation (10) explicitly indicates the dependence of the decision variables $\{\mathbf{e}^t,\mathbf{p}^t,\mathbf{u}^t,\mathbf{v}^t\}$ on the realization of $\boldsymbol{s}^t$.

Let $\mathcal{F}^t$ denote the set of $\{\mathbf{e}^t,\mathbf{p}^t,\mathbf{u}^t,\mathbf{v}^t\}$ satisfying constraints (1) and (2) per t, while $\lambda^i_{kn,1}$ and $\lambda^i_{kn,2}$ denote the Lagrange multipliers associated with the constraints (10d) and (10e). With a convenient notation $\boldsymbol{\lambda}:=\{\lambda^i_{kn,1},\lambda^i_{kn,2},\forall i,k,n\}$, the partial Lagrangian function of (10) is given by

$$L(\mathcal{X},\boldsymbol{\lambda}):=\mathbb{E}[L^t(\mathbf{x}^t,\boldsymbol{\lambda})] \tag{11}$$

where the instantaneous Lagrangian is given by

$$\begin{aligned}L^t(\mathbf{x}^t,\boldsymbol{\lambda}):=\Phi^t(\mathbf{x}^t)&+\sum_{i,k,n}\lambda^i_{kn,1}(t)(\sum_{a\in\mathcal{N}}u^i_{k,an}(t)\\&+\sum_{c\in\mathcal{N}}v^i_{k,cn}(t)+R^{i,t}_{kn}-\sum_{b\in\mathcal{N}}u^i_{k,nb}(t)-p^i_{kn}(t)e_{kn}(t))\\&+\sum_{i,k,k',n}\lambda^i_{kn,2}(t)(p^i_{k'n}(t)e_{k'n}(t)-\sum_{d\in\mathcal{N}}v^i_{k,nd}(t)).\end{aligned} \tag{12}$$

Notice that the instantaneous objective $\Phi^t(\mathbf{x}^t)$ and the instantaneous constraints associated with $\boldsymbol{\lambda}$ are parameterized by the observed state $\boldsymbol{s}^t$ at time t; thus the instantaneous Lagrangian can be written as $L^t(\mathbf{x}^t,\boldsymbol{\lambda})=L(\mathcal{X}(\boldsymbol{s}^t),\boldsymbol{\lambda};\boldsymbol{s}^t)$, and $L(\mathcal{X},\boldsymbol{\lambda})=\mathbb{E}[L(\mathcal{X}(\boldsymbol{s}^t),\boldsymbol{\lambda};\boldsymbol{s}^t)]$.

As a result, the Lagrange dual function is given by

$$D(\boldsymbol{\lambda}):=\min_{\{\mathbf{x}^t\in\mathcal{F}^t\}_t}L(\mathcal{X},\boldsymbol{\lambda}), \tag{13}$$

and the dual problem of (9) is: $\max_{\boldsymbol{\lambda}\ge 0}\ D(\boldsymbol{\lambda})$, where " $\ge$ " is defined entry-wise.

For the dual problem, we can take a standard gradient method to obtain the optimal $\boldsymbol{\lambda}^*$ [32]. This amounts to running the following iterations slot by slot

$$\lambda^i_{kn,1}(t+1)=[\lambda^i_{kn,1}(t)+\epsilon g_{\lambda^i_{kn,1}}(t)]^+,\quad \forall i,k,n, \tag{14a}$$

$$\lambda^i_{kn,2}(t+1)=[\lambda^i_{kn,2}(t)+\epsilon g_{\lambda^i_{kn,2}}(t)]^+,\quad \forall i,k,n. \tag{14b}$$

where $\epsilon>0$ is an appropriate stepsize. The gradient $\boldsymbol{g}(t):=[g_{\lambda^i_{kn,1}}(t),g_{\lambda^i_{kn,2}}(t),\forall i,k,n]$ can be expressed as

$$g_{\lambda^i_{kn,1}}(t)=\mathbb{E}[\sum_{a\in\mathcal{N}}u^i_{k,an}(t)+\sum_{c\in\mathcal{N}}v^i_{k,cn}(t)+R^{i,t}_{kn} - \sum_{b\in\mathcal{N}}u^i_{k,nb}(t)-p^i_{kn}(t)e_{kn}(t)], \tag{15a}$$

$$g_{\lambda^i_{kn,2}}(t)=\mathbb{E}[p^i_{k'n}(t)e_{k'n}(t)-\sum_{d\in\mathcal{N}}v^i_{k,nd}(t)], \tag{15b}$$

where $\mathbf{x}^t:=\{\mathbf{e}^t,\mathbf{p}^t,\mathbf{u}^t,\mathbf{v}^t\}$ is given by

$$\mathbf{x}^t=\arg\min_{\mathbf{x}^t\in\mathcal{F}^t}L^t(\mathbf{x}^t,\boldsymbol{\lambda}). \tag{16}$$

Note that a challenge associated with (15) is sequentially taking expectations over the random vector $\boldsymbol{s}^t$ to compute the gradient $\boldsymbol{g}(t)$. This would require high-dimensional integration over an unknown probabilistic distribution function of $\boldsymbol{s}^t$; or equivalently, computing the corresponding time-averages over an infinite time horizon. Such a requirement is impractical since the computational complexity could be prohibitively high.

To bypass this impasse, we propose to rely on a stochastic dual gradient approach, which is able to combat randomness in the absence of the a-priori knowledge on the statistics of variables. Stochastic gradient descent method is applied to minimize the expectation of the instantaneous cost of NFV at each time slot, subject to time-varying

random traffic demand, link bandwidth, and processor capacity. Stochastic gradient descent method is a stochastic approximation of gradient descent method. It can recursively and asymptotically converge to the neighborhood of the global optimum in the presence of time-varying random parameters. Specifically, dropping $\mathbb{E}$ from (15), we propose the following iterations

$$\tilde{\lambda}^i_{kn,1}(t+1) = \tilde{\lambda}^i_{kn,1}(t) + \epsilon[\sum_{a\in\mathcal{N}} u^i_{k,an}(t) + \sum_{c\in\mathcal{N}} v^i_{k,cn}(t) + R^{i,t}_{kn} - \sum_{b\in\mathcal{N}} u^i_{k,nb}(t) - p^i_{kn}(t)e_{kn}(t)]^+, \quad (17a)$$

$$\tilde{\lambda}^i_{kn,2}(t+1) = \tilde{\lambda}^i_{kn,2}(t) + \epsilon[p^i_{k'n}(t)e_{k'n}(t) - \sum_{d\in\mathcal{N}} v^i_{k,nd}(t)]^+, \quad (17b)$$

where $\tilde{\boldsymbol{\lambda}}^t = \{\tilde{\lambda}^i_{kn,1}(t), \tilde{\lambda}^i_{kn,2}(t), \forall i,k,n\}$ collects the stochastic estimates of the variables in (14), and $\mathbf{x}^t(\tilde{\boldsymbol{\lambda}})$ is obtained by solving (16) with $\boldsymbol{\lambda}$ replaced by $\tilde{\boldsymbol{\lambda}}^t$, $\forall i,k,n$.

Note that the interval of updating (17) coincides with slots. In other words, the update of (17) is an *online* approximation of (14) based on the *instantaneous* decisions $\mathbf{x}^t(\tilde{\boldsymbol{\lambda}}^t)$ per slot t. This stochastic approach becomes possible due to the decoupling of optimization variables over time in (9).

Relying on the so-called Lyapunov optimization technique in [29], we can formally establish that:

Theorem 1. *If $\boldsymbol{s}^t$ is i.i.d. over slots, then the time-average cost of* (10) *with the multipliers updated by* (17) *satisfies*

$$\Phi^* \le \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\Phi^t(\mathbf{x}^t))\right] \le \Phi^* + \epsilon B \quad (18a)$$

where $B = \frac{9}{2}(N^{\max} l^{\max})^2 + \frac{3}{2}(R^{\max 2} + p^{\max 2})$, $N^{\max}$ is the maximum degree of VMs, $l^{\max} = \max_{[a,b]} l^{\max}_{ab}$ and $p^{\max} = \max_n p^{\max}_n$; Φ^* is the optimal value of (7) *under any feasible control policy (i.e., the processing and routing decisions per VM), even if that relies on knowing future realizations of random variables.*

Assume that there exists a stationary policy $\mathcal{X}$ and $\mathbb{E}[\sum_{a\in\mathcal{N}} u^i_{k,an}(t) + \sum_{c\in\mathcal{N}} v^i_{k,cn}(t) + R^{i,t}_{kn} - \sum_{b\in\mathcal{N}} u^i_{k,nb}(t) - p^i_{kn}(t)e_{kn}(t)] \le -\zeta$, and $\mathbb{E}[p^i_{k'n}(t)e_{k'n}(t) - \sum_{d\in\mathcal{N}} v^i_{k,nd}(t)] \le -\zeta$, where $\zeta > 0$ is a slack vector constant. Then all queues are stable, and the time-average queue length satisfies:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{i,k,n} \mathbb{E}[Q^i_{kn}(t) + q^i_{kn}(t)] = \mathcal{O}(\frac{1}{\epsilon}). \quad (18b)$$

Proof. See Appendices A and B. □

Theorem 1 asserts that the time-average cost of (10) obtained by the stochastic dual gradient approach converges to an $\mathcal{O}(\epsilon)$ neighborhood of the optimal solution, where the region of neighborhood vanishes as the stepsize $\epsilon \to 0$ (or in other words, the convergence time is $\mathcal{O}(1/\epsilon)$). The typical tradeoff from the stochastic network optimization holds in this case [29]: an $\mathcal{O}(1/\epsilon)$ queue length is necessary, when an $\mathcal{O}(\epsilon)$ close-to-optimal cost is achieved. Different from [29], here the Lagrange dual theory is utilized to simplify the arguments, as shown in Appendices A and B.

The asymptotic approximation of the proposed distributed online scheme to the cost lower bound achieved offline in a posterior manner is rigorously proved. The lower bound corresponds to the assumption that all the randomnesses are precisely known in prior and the optimal decisions over infinite time-horizon are all derived. This lower bound would violate causality and be computationally prohibitive to achieve, even in an offline fashion, given an infinite number of variables. Theorem 1 indicates that the proposed scheme can increasingly approach the lower bound by increasing the tolerance to queue backlogs or delays.

The size of services does not violate the asymptotic optimality of the proposed algorithm. The reason is that the processing and routing decisions $\mathbf{x}^t$ need to be discretized to the numbers of network services processed or delivered, i.e., $S_i\lfloor\frac{\mathbf{x}^t}{S_i}\rfloor$, in practice. Here, S_i denotes the size of a type-i service. Let $S^{\max}$ denote the maximum size of services. According to (6), the difference between the cost under $\mathbf{x}^t$ and the cost under $S_i\lfloor\frac{\mathbf{x}^t}{S_i}\rfloor$ is at most $(\alpha^t_n + \beta^t_{[a,b]})S_i^2 \le (\alpha^t_n + \beta^t_{[a,b]})(S^{\max})^2$. The right-hand side (RHS) of the inequality is independent of ϵ, and hence it is $O(1)$ and becomes part of the constant B in (18a) in Theorem 1. The asymptotic optimality in Theorem 1 is preserved.

3.2 Distributed online implementation

The dual iteration (17) coincides with (3) and (4) for $\tilde{\lambda}^i_{kn,1}(t)/\epsilon = Q^i_{kn}(t)$ and $\tilde{\lambda}^i_{kn,2}(t)/\epsilon = q^i_{kn}(t), \forall i,k,n,t$; this can be interpreted by using the concept of virtual queue of this parallelism [29]. With $\tilde{\lambda}^i_{kn,1}(t)$ substituted by $\epsilon Q^i_{kn}(t)$ and $\tilde{\lambda}^i_{kn,2}(t)$ substituted by $\epsilon q^i_{kn}(t)$, we can obtain the desired $\mathbf{x}^t(\mathbf{A}(t))$ by solving the following problem:

$$\min_{\mathbf{x}^t\in\mathcal{F}^t} \frac{1}{\epsilon}\Phi^t(\mathbf{x}^t) + \sum_{i,k,n} Q^i_{kn}(t)[\sum_{a\in\mathcal{N}} u^i_{k,an}(t) + \sum_{c\in\mathcal{N}} v^i_{k,cn}(t) + R^{i,t}_{kn} - \sum_{b\in\mathcal{N}} u^i_{k,nb}(t) - p^i_{kn}(t)e_{kn}(t)] + \sum_{i,k,k',n} q^i_{kn}(t)[p^i_{k'n}(t)e^i_{k'n}(t) - \sum_{d\in\mathcal{N}} v^i_{k,nd}(t)]. \quad (19)$$

Through rearrangement, (19) is equivalent to

$$\min_{\mathbf{x}^t\in\mathcal{F}^t} \sum_{i,k,n,b\in\mathcal{N}} [f_1(\mathbf{e}^t,\mathbf{p}^t) + f_2(\mathbf{u}^t) + f_3(\mathbf{v}^t)] \quad (20)$$

where

$$f_1(\mathbf{e}^t,\mathbf{p}^t) = [\frac{\alpha^t_n}{\epsilon}(p^i_{kn}(t))^2 - (Q^i_{kn}(t) - q^{i}_{k''n}(t))p^i_{kn}(t)]e_{kn}(t); \quad (21a)$$

$$f_2(\mathbf{u}^t) = [\frac{\beta^t_{[n,b]}}{\epsilon}(u^i_{k,nb}(t))^2 - (Q^i_{kn}(t) - Q^i_{kb}(t))u^i_{k,nb}(t); \quad (21b)$$

$$f_3(\mathbf{v}^t) = \frac{\beta^t_{[n,b]}}{\epsilon}(v^i_{k,nb}(t))^2 - (q^i_{kn}(t) - Q^i_{kb}(t))v^i_{k,nb}(t). \quad (21c)$$

Here, $f_{k''}$ denotes the VNF to be processed after f_k for type-i network services.

Problem (20) can be readily solved by decoupling between $e_{kn}(t)$, $p^i_{kn}(t)$, $u^i_{k,nb}(t)$ and $v^i_{k,nb}(t)$, and between the VMs. Specifically, (20) can be decoupled into the following

subproblems per VM or per inter-VM link:

$$\min_{\mathbf{e}^t,\mathbf{p}^t} \quad f_1(\mathbf{e}^t;\mathbf{p}^t), \tag{22a}$$

$$\min_{\mathbf{u}^t} \quad f_2(\mathbf{u}^t); \tag{22b}$$

$$\min_{\mathbf{v}^t} \quad f_3(\mathbf{v}^t). \tag{22c}$$

Problem (22a) is a mixed integer programming. Its solution can be obtained by comparing the minimums of $f_1(\mathbf{e}^t,\mathbf{p}^t)$ separately achieved under $e_{kn}(t)=0$ and 1. In the case of $e_{kn}(t)=0$, $f_1(\mathbf{e}^t,\mathbf{p}^t)=0$. In the case of $e_{kn}(t)=1$, (22a) becomes the minimization of a quadratic function of $p^i_{kn}(t)$, where the optimal solution is given by

$$p^i_{kn}{}^*(t)=\min\{\max\{\frac{\epsilon(Q^i_{kn}(t)-q^i_{k''n}(t))}{2\alpha^t_n},0\},p^{\max}_n\},\ \forall i,k. \tag{23}$$

Then, the optimal objective of (22a) can be obtained by substituting (23) into $f_1(\mathbf{e}^t,\mathbf{p}^t)$, as given by

$$P^i_{kn}:=\begin{cases}-\frac{\epsilon(Q^i_{kn}(t)-q^i_{k''n}(t))^2}{4\alpha^t_n}, & \text{if } Q^i_{kn}(t)-q^i_{k''n}(t)>0;\\ 0, & \text{if } Q^i_{kn}(t)-q^i_{k''n}(t)\le 0.\end{cases} \tag{24}$$

Since every VM only runs a single VNF (i.e., $\sum_k e_{kn}(t)=1,\forall n,t$), we have

$$e_{kn}{}^*(t)=\begin{cases}1, & \text{if } k=\arg\min_k P^i_{kn};\\ 0, & \text{otherwise.}\end{cases} \tag{25}$$

Problems (22b) and (22c) are the minimizations of quadratic functions of $\mathbf{u}^t$ and $\mathbf{v}^t$, respectively. Like (22a) under $e_{kn}(t)=1$, the optimal solutions for (22b) and (22c) are

$$u^i_{k,nb}{}^*(t)=\min\{\max\{\frac{\epsilon(Q^i_{kn}(t)-Q^i_{kb}(t))}{2\beta^t_{[n,b]}},0\},l^{\max}_{ab}\},\ \forall i,k;$$

$$v^i_{k,nb}{}^*(t)=\min\{\max\{\frac{\epsilon(q^i_{kn}(t)-Q^i_{kb}(t))}{2\beta^t_{[n,b]}},0\},l^{\max}_{ab}\},\ \forall i,k; \tag{26}$$

with their corresponding objectives given by

$$U^i_{k,nb}:=\begin{cases}-\frac{\epsilon(Q^i_{kn}(t)-Q^i_{kb}(t))^2}{4\beta^t_{[n,b]}}, & \text{if } Q^i_{kn}(t)-Q^i_{kb}(t)>0;\\ 0, & \text{if } Q^i_{kn}(t)-Q^i_{kb}(t)\le 0;\end{cases}$$

$$V^i_{k,nb}:=\begin{cases}-\frac{\epsilon(q^i_{kn}(t)-Q^i_{kb}(t))^2}{4\beta^t_{[n,b]}}, & \text{if } q^i_{kn}(t)-Q^i_{kb}(t)>0;\\ 0, & \text{if } q^i_{kn}(t)-Q^i_{kb}(t)\le 0.\end{cases} \tag{27}$$

At each slot, a VM can prioritize the queues of different service types to be processed by different VNFs, and process or route services from the queue with the highest priority. The priority is ranked based on the objectives P^i_{kn}, $U^i_{k,nb}$ and $V^i_{k,nb}$ in (24) and (27). For this reason, we refer to P^i_{kn}, $U^i_{k,nb}$ and $V^i_{k,nb}$ as queue-price objectives. The processing and routing decisions can be made by one-to-one mapping between the queues and outgoing links/processor to minimize the total of selected non-zero objectives, as summarized in Algorithm 1.

The optimal strategy each VM takes does not stop different network services from concurrently occupying a link. For example, if the most cost-effective service type for a link is not sufficient to occupy the entire link, the second-to-most cost-effective service type is selected to occupy the rest part of the link. The link can be bidirectional. The different service types can travel in the opposite directions.

Algorithm 1 Distributed Online Optimization of NFV

1: **for** $t=1,2\ldots$ **do**
2: Each VM n observes the queue lengths of its own and its one-hop neighbors.
3: Install VNF f_k at VM n based on (25).
4: Repeatedly send network services to the VM processor or outgoing links with the minimum non-zero queue-price objectives in (24) and (27), using the optimal rates derived in (23) and (26), until either the processor and all outgoing links are scheduled or the remaining objectives are all zero.
5: Update $Q^i_{kn}(t)$ and $q^i_{kn}(t)$ for all nodes and services via the dynamics (3) and (4).
6: **end for**

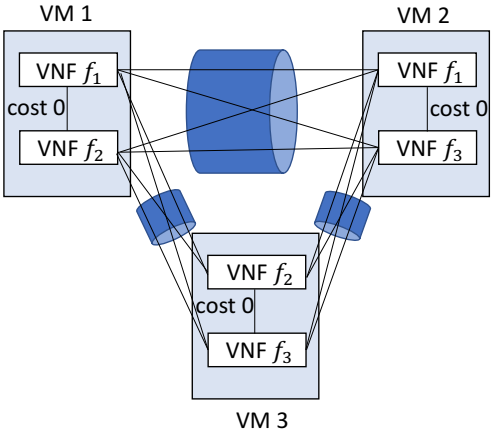


Fig. 2. An illustration on VMs running multiple VNFs, where VNFs can be interpreted as "VMs" and VMs can be interpreted as "clusters of VMs." Then the VM-based online optimization developed in this paper can be readily applied to the "VMs."

Note that Algorithm 1 is decentralized, since every VM only needs to know the queue lengths of its own and its immediate neighbors. Optimal decisions of a VM, locally made by comparing the queue-price objectives, comply with (17) and therefore preserve the asymptotic optimality of the entire network, as dictated in Theorem 1. As per timeslot, the computational complexity consists of computing $(|\mathcal{I}|KN+2|\mathcal{I}|KN^2)$ queue-price objectives and their corresponding optimal processing and routing rates for all VMs. Therefore, the proposed algorithm has a computational complexity of $\mathcal{O}(|\mathcal{I}|KN^2)$. With this quadratic complexity in terms of network size, Algorithm 1 is scalable and suitable to practical scenarios of large-scale network platforms.

Algorithm 1 can be readily extended to general scenarios where a VM runs multiple VNFs; see Fig. 2. In this case, all VNFs can be first interpreted as separate "VMs" in the context of the baseline case of a VNF per VM, and colocated VNFs at a VM then become a cluster of multiple "VMs."

No cost incurs on the connections between the "VMs" within the cluster. The only difference from the baseline scenario of a VNF per VM, as described in Algorithm 1, is that, between the clusters, only a pair of "VMs" which are respectively from the two clusters and the most cost-effective to transmit workloads, can be activated. This can be achieved by comparing the price weights of the links to pick up the most cost-effective link between clusters. The optimal decisions of processing at each of the "VMs" stay unchanged.

Algorithm 1 can readily incorporate the case where there are multiple co-located VMs at a host server. Specifically, the routing cost between these VMs can be set to zero, and the bandwidth of the virtual links between the VMs can be high. Hence, tasks can be offloaded between the VMs to balance the backlogs for the VMs between the same host server. The centralized decisions of a host server for the VMs executed on it would not improve the performance of the algorithm. As a matter of fact, the centralized decisions of a host server would be exactly the same as the distributed decisions of the VMs executed on the server. This is because Algorithm 1 first suppresses the temporal coupling of the optimal decisions by applying the stochastic gradient descent method (leading to the asymptotic optimality), and then decouples the asymptotically optimal decisions at every time slot among individual VMs. The decoupling of the asymptotically optimal decisions among the VMs is loss-free at every time slot; see (17), (19), and (22). To this end, even if all the VMs are co-located at the same server, the centralized decisions of the host server at every time slot would be the same as the decoupled decisions of the VMs which are made in a decentralized manner.

Further, admission control is not considered at the VMs or servers, but can be straightforwardly incorporated in the proposed approach. In this case, the amount of requests that can be admitted to a VM at a time slot is no larger than the amount of traffic processed at the VM or offloaded to other VMs at the time slot, thereby preserving the stability of the VM and the entire platform. The admission control does not invalidate the asymptotic optimality of the proposed approach, as it is based on, and does not violate, the adaptive replacement of VNF and allocation of resources optimized in the approach.

4 Optimal Placement and Operation of NFV at Different Timescales

In this section, we consider a more practical scenario where the placement of VNFs is carried out at the VMs at a much larger time interval, i.e., at time $\tau = mT_\Delta (m = 1, 2, \ldots)$, rather than on a slot basis. This is because the installation of VNFs at the VMs could cause interruptions to network service provisions. We prove that if the placement and the operation of NFV are jointly optimized at two different timescales, the aforementioned asymptotic optimality of the proposed approach can be preserved.

4.1 Two-timescale placement and operation

By evaluating (22) at two different timescales, the placement of VNFs, and the processing and routing of network services, can be carried out as follows:

- *Placement of VNFs at a T_Δ-slot interval:* At time slot $\tau = mT_\Delta$, each VM n decides on the VNF to install to minimize the expectation of the sum of $f_1(\mathbf{e}^t, \mathbf{p}^t)$ in (22a) over the time window $t = \{\tau, \ldots, \tau + T_\Delta - 1\}$, i.e., $\mathbb{E}\{\sum_{t=\tau}^{\tau+T_\Delta-1} f_1(\mathbf{e}^t, \mathbf{p}^t)\}$, as given by

$$\min_{\mathbf{e}^t} \mathbb{E}\Big\{ \sum_{t=\tau}^{\tau+T_\Delta-1} \sum_{i,k} [\frac{\alpha_n^t}{\epsilon}(p_{kn}^i(t))^2 - (Q_{kn}^i(t) - q_{k''n}^i(t))p_{kn}^i(t)]e_{kn}(\tau)\Big\}. \quad (28)$$

- *Processing and routing of network services per slot t:* Per slot t, each VM processes and routes network services, following Algorithm 1, given the placement decisions of VNFs given by (28).
- *Queue update:* Each VM updates its queues $Q_{kn}^i(t)$ and $q_{kn}^i(t)$ at every slot t based on (3) and (4).

Note that the optimal solutions to (28) require future knowledge on service arrivals $\{R_{kn}^{i,t}, t = \tau, \ldots, \tau + T_\Delta - 1\}$, and the prices of service processing and routing $\{\alpha_n^t, \beta_{[a,b]}^t, t = \tau, \ldots, \tau + T_\Delta - 1\}$. This would violate causality. We propose to take an approximation by setting the future queue backlogs as their current backlogs at slot $\tau = mT$, as given by

$$\hat{Q}_{kn}^i(t) = Q_{kn}^i(\tau); \quad (29a)$$

$$\hat{q}_{kn}^i(t) = q_{kn}^i(\tau),\ \forall t = \tau, \ldots, \tau + T_\Delta - 1, \quad (29b)$$

where $\hat{Q}_{kn}^i(t)$ and $\hat{q}_{kn}^i(t)$ are the approximated queue backlogs. Taking the approximation of (29) and that the stochastic variables $\boldsymbol{s}^t := [R_{kn}^{i,t}, \alpha_n^t, \beta_{[a,b]}^t, \forall [a,b], i, k, n]$ to be invariant in the coming time window, (28) can be reduced to the per-slot problem, as given in (22a). Therefore, as per time slot $\tau = mT_\Delta$, the VNFs can be installed at the VMs based on (25), as summarized in Algorithm 1.

4.2 Optimality loss of VNF placement

We next analyze the optimality loss due to the approximation of (29). We can prove that the optimality loss is bounded and does not compromise the asymptotic optimality of the proposed approach. To improve tractability, an upper bound of the optimality loss is evaluated in the case where all the variables $\{\mathbf{e}^t, \mathbf{p}^t, \mathbf{u}^t, \mathbf{v}^t\}$ are optimized under the assumption of the availability of the full knowledge on the future T_Δ time slots, rather than taking the approximation of (29).

Based on (29), we first establish the following lemma:

Lemma 1. *At any slot t, the differences between the approximated and actual queue backlogs in* (29) *are bounded by*

$$\left|Q_{kn}^i(t) - \hat{Q}_{kn}^i(t)\right| \le T_\Delta \omega_Q, \quad (30a)$$

$$\left|q_{kn}^i(t) - \hat{q}_{kn}^i(t)\right| \le T_\Delta \omega_q, \quad (30b)$$

where the constants $\omega_Q = \max\{N^{\max} l^{\max} + p^{\max}, 2N^{\max} l^{\max} + R^{\max}\}$, and $\omega_q = \max\{N^{\max} l^{\max}, p^{\max}\}$.

Proof. For any two consecutive slots t and $t+1$, the difference of the queue backlogs is bounded, i.e., $\left|Q_{kn}^i(t+1) - Q_{kn}^i(t)\right| \le \omega_Q$, where ω_Q is the maximum difference between the departure and the arrival of $\mathbf{Q}(t)$, denoted

by $\max\{N^{\max}l^{\max}+p^{\max}, 2N^{\max}l^{\max}+R^{\max}\}$. According to (29) and the inequality $|a+b| \leq |a|+|b|$, we have $\left|Q^i_{kn}(t)-\hat{Q}^i_{kn}(t)\right| = \left|Q^i_{kn}(t)-Q^i_{kn}(\tau)\right| = \left|\sum_{t_0=\tau}^{t}[Q(t_0+1)-Q(t_0)]\right| \leq (t-\tau)\omega_Q \leq T_\Delta\omega_Q$, where $\tau = mT_\Delta$ and $t = \tau,\ldots,\tau+T_\Delta-1$. Therefore, (30a) is proved. Likewise, (30b) can be proved. □

Based on Lemma 1, we are ready to obtain the following theorem:

Theorem 2. *The optimality loss of the solution for* (20) *due to the approximation of the queue backlogs in* (29) *is bounded, i.e.,*

$$\Big|\sum_{i,k,n,b\in\mathcal{N}}[f_1(\hat{\mathbf{e}}^t,\hat{\mathbf{p}}^t)+f_2(\hat{\mathbf{u}}^t)+f_3(\hat{\mathbf{v}}^t)] - \sum_{i,k,n,b\in\mathcal{N}}[f_1(\mathbf{e}^t,\mathbf{p}^t)+f_2(\mathbf{u}^t)+f_3(\mathbf{v}^t)]\Big| \leq \epsilon C, \quad (31)$$

where $C := \epsilon {T_\Delta}^2 N^2|\mathcal{I}|K[(\frac{1}{2\alpha^{\max}}+\frac{1}{2\beta^{\max}})(\omega_Q+\omega_q)^2 + \frac{2}{\beta^{\max}}\omega_Q^2]$; $\alpha^{\max}=\max_{n,t}\alpha^t_n$, *and* $\beta^{\max}=\max_{[a,b],t}\beta^t_{[a,b]}$.

Proof. See Appendix C. □

We can see from Theorem 2 that the optimality loss of the two-timescale control increases quadratically with the time interval of VNF placement, T_Δ. This allows us to balance between the optimality loss and the cost of VNF placement. As dictated in Theorems 1 and 2, the total optimality loss of the two-timescale approach for problem (7) is upper bounded, as given by

$$\overline{\Phi(\hat{\mathbf{x}}^t)} \leq \Phi^* + \epsilon(B+C), \quad (32)$$

where $\overline{\Phi(\hat{\mathbf{x}}^t)}$ is the time-average cost under the two-timescale approach. In other words, the two-timescale placement and operation of NFV preserves the asymptotic optimality with approximated queue backlogs.

4.3 Impact of non-negligible deployment costs

It is proved in Section 4.2 that the multi-timescale design preserves the asymptotic optimality in the case where a charge of VNF only takes a portion of a time slot and the new VNF can start to function straight away after installation (in other words, the VM does not suspend operations for the installation). In this section, we proceed to consider a further generalized case where the replacement of a VNF can take a number of time slots (e.g., T_{null} slots), and the operations of the VM (including transmission and processing) are suspended during the time slots. In this case, the tradeoff of $[\mathcal{O}(\epsilon),\mathcal{O}(\frac{1}{\epsilon})]$ remains valid. This is proved by considering the worst-case scenario in terms of the time-average cost of the platform, where every VM changes its VNF every T_Δ slots. In other words, in T_{null} slots out of every T_Δ slots, the entire platform is suspended. By extending (18) and (32), we can show that the time-average cost of the platform is upper-bounded by $\frac{T_\Delta-T_{\text{null}}}{T_\Delta}\mathcal{O}(\epsilon)$, i.e.,

$$\overline{\Phi'(\hat{\mathbf{x}}^t)} \leq \frac{T_\Delta-T_{\text{null}}}{T_\Delta}[\Phi^*+\epsilon(B+C)], \quad (33)$$

where $\overline{\Phi'(\hat{\mathbf{x}}^t)}$ is the time-average cost under the two-timescale approach with non-negligible deployment cost. In the worst-case scenario, the time-average queue length of the platform grows to $\frac{T_\Delta}{T_\Delta-T_{\text{null}}}\mathcal{O}(\frac{1}{\epsilon})$, i.e.,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i,k,n}\mathbb{E}[Q^i_{kn}(t)+q^i_{kn}(t)] = \frac{T_\Delta}{T_\Delta-T_{\text{null}}}\mathcal{O}(\frac{1}{\epsilon}). \quad (34)$$

From (33) and (34), there is still an $[\mathcal{O}(\epsilon),\mathcal{O}(\frac{1}{\epsilon})]$-tradeoff between the optimality loss and queue length in the generalized case where the replacement of a VNF can take multiple time slots.

4.4 Impact of non-negligible propagation delays

The asymptotic optimality of the proposed two-timescale approach is not compromised in the presence of non-negligible propagation delays between neighboring VMs connected through multiple network nodes.

We assume that the knowledge of a VM on the queues of its neighboring VM n is outdated by ω time slots (due to the existence of network nodes between the VMs). We have $\omega \leq \omega_{\max} \leq N'-1$, where $\omega_{\max}$ is the maximum propagation delay measured in hops between any pair of nodes, and N' is the number of nodes in the network. Let $\tilde{Q}^i_{kn}(t)$ and $\tilde{q}^i_{kn}(t)$ denote the outdated queue lengths that a neighboring VM of VM n has on VM n's tasks at time slot t. $\tilde{Q}^i_{kn}(t)$ and $\tilde{q}^i_{kn}(t)$ are outdated by ω slots, as compared to the actual queues at the time slot, $Q^i_{kn}(t)$ and $q^i_{kn}(t)$. By referring to Lemma 1, both the propagation delays and multi-timescale operations, leading to the outdated queue knowledge at neighboring VMs, cause a bounded difference between the outdated and up-to-date queue backlogs per slot t, as given by

$$\left|Q^i_{kn}(t)-\tilde{Q}^i_{kn}(t)\right| \leq (T_\Delta+\omega_{\max})\omega_Q, \quad (35a)$$
$$\left|q^i_{kn}(t)-\tilde{q}^i_{kn}(t)\right| \leq (T_\Delta+\omega_{\max})\omega_q, \quad (35b)$$

where $\omega_{\max}\omega_Q$ and $T_\Delta\omega_Q$ account for the maximum outdated backlogs due to the non-negligible propagation delays and multi-timescale operations, respectively.

The outdated knowledge of VM n on its neighboring VM b's queues can affect its decision of service routing over link $[n,b]$ at time slot t. By substituting (35) into (21) and then applying Theorem 2, we can show that the optimality loss of the two-timescale solution for (20) under non-negligible propagation delays can be updated by

$$\Big|\sum_{i,k,n,b\in\mathcal{N}}[f_1(\hat{\mathbf{e}}^t,\hat{\mathbf{p}}^t)+f_2(\tilde{\mathbf{u}}^t)+f_3(\tilde{\mathbf{v}}^t)] - \sum_{i,k,n,b\in\mathcal{N}}[f_1(\mathbf{e}^t,\mathbf{p}^t)+f_2(\mathbf{u}^t)+f_3(\mathbf{v}^t)]\Big| \leq \epsilon(C+D), \quad (36)$$

where $D := N^2|\mathcal{I}|K(2T_\Delta\omega_{\max}+\omega^2_{\max})[\frac{2}{\beta^{\max}}\omega_Q^2 + \frac{1}{2\beta^{\max}}(\omega_Q+\omega_q)^2]$. The functions $f_2(\hat{\mathbf{u}}^t)$ and $f_3(\hat{\mathbf{v}}^t)$ in Theorem 2 is updated by $f_2(\tilde{\mathbf{u}}^t)$ and $f_3(\tilde{\mathbf{v}}^t)$; the function $f_1(\hat{\mathbf{e}}^t,\hat{\mathbf{p}}^t)$ is unaffected in the presence of non-negligible propagation delays, since VM n always has the up-to-date backlogs of its own queues. The optimality loss is also upper bounded, and asymptotically diminishes as the stepsize ϵ decreases.

Algorithm 2 Distributed Learn-and-Adapt NFV Optimization

1: **for** $t = 1, 2 \dots$ **do**
2: **Online processing and routing (1st gradient):**
3: Construct the effective dual variable via (39), observe the current state $\boldsymbol{s}^t$, and obtain placement, processing and routing decisions $\mathbf{x}^t(\boldsymbol{\gamma}^t)$ by minimizing online Lagrangian (38).
4: Update the instantaneous queue length $\mathbf{Q}(t+1)$ and $\mathbf{q}(t+1)$ with $\mathbf{x}^t(\boldsymbol{\gamma}^t)$ via queue dynamics (3) and (4).
5: **Statistical learning (2nd gradient):**
6: Obtain variable $\mathbf{x}^t(\hat{\boldsymbol{\lambda}}^t)$ by solving online Lagrangian minimization with sample $\boldsymbol{s}^t$ via (41).
7: Update the empirical dual variable $\hat{\boldsymbol{\lambda}}^{t+1}$ via (40).
8: **end for**

Based on Theorem 1 and (36), the optimality loss of the two-timescale approach under non-negligible propagation delays is upper bounded by

$$\overline{\Phi(\tilde{\mathbf{x}}^t)} \leq \Phi^* + \epsilon(B + C + D), \tag{37}$$

where $\overline{\Phi(\tilde{\mathbf{x}}^t)}$ is the time-average cost of the two-timescale approach under non-negligible propagation delays, and it can asymptotically approach the global optimum Φ^* as ϵ decreases. The asymptotic optimality of the proposed approach is proved under non-negligible propagation delays.

4.5 Learn-and-adapt for placement acceleration

While preserving the asymptotic optimality, the larger timescale can slow down the optimal placement of VNFs. We propose to speed the placement up through a learning and adaptation method [31]. The Lagrange multipliers $\tilde{\lambda}^i_{kn,1}(t)$ and $\tilde{\lambda}^i_{kn,2}(t)$ play the key roles in the proposed distributed online optimization of NFV in (17). We can incrementally learn these Lagrange multipliers from observed data and speed up the convergence of the multipliers driven by the learning process.

In the proposed learn-and-adapt scheme, with the online learning of $\tilde{\lambda}^i_{kn,1}(t)$ and $\tilde{\lambda}^i_{kn,2}(t), \forall n, i$ at each slot t, two stochastic gradients are updated using the current $\boldsymbol{s}^t$. The first gradient $\boldsymbol{\gamma}^t$ is designed to minimize the instantaneous Lagrangian for optimal decision makings on processing or routing network services, as given by [cf. (16)]

$$\mathbf{x}^t(\boldsymbol{\gamma}^t) = \arg\min_{\mathbf{x}^t \in \mathcal{F}^t} L^t(\mathbf{x}^t, \boldsymbol{\gamma}^t) \tag{38}$$

which depends on what we term *effective* multiplier $\boldsymbol{\gamma}^t := \{\gamma^i_{kn,1}(t), \gamma^i_{kn,2}(t), \forall n, i\}$, as given by

$$\underbrace{\boldsymbol{\gamma}^t}_{\text{effective multiplier}} = \underbrace{\hat{\boldsymbol{\lambda}}^t}_{\text{statistical learning}} + \underbrace{\epsilon \mathbf{A}(t) - \boldsymbol{\theta}}_{\text{online adaptation}}, \tag{39}$$

where $\hat{\boldsymbol{\lambda}}^t := \{\hat{\lambda}^i_{kn,1}(t), \hat{\lambda}^i_{kn,2}(t), \forall i, k, n\}$ is the empirical dual variable, and $\boldsymbol{\theta}$ controls the bias of $\boldsymbol{\gamma}^t$ in the steady state, and can be judiciously designed to achieve the improved cost-delay tradeoff, as will be shown in Theorem 3.

For a better illustration of the effective multiplier in (39), we call $\hat{\boldsymbol{\lambda}}(t)$ the statistically learnt dual variable to obtain the exact optimal argument of the dual problem $\max_{\boldsymbol{\lambda} \succeq 0} D(\boldsymbol{\lambda})$. We call $\epsilon \mathbf{A}(t)$ (which is exactly $\boldsymbol{\lambda}$ as shown in (17)) the online adaptation term, since it can track the instantaneous change of system statistics. The control variable ϵ tunes the weights of these two factors.

The second gradient is designed to simply learn the stochastic gradient of (13) at the previous empirical dual variable $\hat{\boldsymbol{\lambda}}^t$, and implement a gradient ascent update as

$$\begin{aligned}
\hat{\lambda}^i_{kn,1}(t+1) = \hat{\lambda}^i_{kn,1}(t) + \eta(t)[&\sum_{a \in \mathcal{N}} u^i_{k,an}(\hat{\lambda}^i_{kn,1}(t)) + R^{i,t}_{kn} \\
&+ \sum_{c \in \mathcal{N}} v^i_{k,cn}(\hat{\lambda}^i_{kn,1}(t)) - \sum_{b \in \mathcal{N}} u^i_{k,nb}(\hat{\lambda}^i_{kn,1}(t)) \\
&- p^i_{kn}(\hat{\lambda}^i_{kn,1}(t)) e_{kn}(\hat{\lambda}^i_{kn,1}(t))]^+, \\
\hat{\lambda}^i_{kn,2}(t+1) = \hat{\lambda}^i_{kn,2}(t) + \eta(t)[&p^i_{k'n}(\hat{\lambda}^i_{k'n,2}(t)) e_{k'n}(\hat{\lambda}^i_{k'n,1}(t)) \\
&- \sum_{d \in \mathcal{N}} v^i_{k,nd}(\hat{\lambda}^i_{kn,2}(t))]^+,
\end{aligned} \tag{40}$$

where $\eta(t)$ is a proper diminishing stepsize, and the "virtual" allocation $\mathbf{x}^t(\hat{\boldsymbol{\lambda}}^t)$ can be found by solving

$$\mathbf{x}^t(\hat{\boldsymbol{\lambda}}^t) = \arg\min_{\mathbf{x}^t \in \mathcal{F}^t} L^t(\mathbf{x}^t, \hat{\boldsymbol{\lambda}}^t). \tag{41}$$

With learn-and-adaption incorporated, Algorithm 2 takes an additional learning step in Algorithm 1, i.e., (40), which adopts gradient ascent with diminishing stepsize $\eta(t)$ to find the "best empirical" dual variable from all observed network states. In the transient stage, the extra gradient evaluations and empirical dual variables accelerate the convergence speed of Algorithm 1; while in the steady stage, the empirical dual variable approaches the optimal multiplier, which significantly reduces the steady-state queue lengths.

Using the learn-and-adapt approach, we are ready to arrive at the following theorem [31, Theorems 2 and 3].

Theorem 3. *Suppose that the assumptions in Theorem 1 are satisfied. Then with $\boldsymbol{\gamma}^t$ defined in* (39) *and $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\epsilon}\log^2(\epsilon))$, Algorithm 2 yields a near-optimal solution for* (7) *in the sense that*

$$\Phi^* \leq \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\Phi^t\left(\mathbf{x}^t(\boldsymbol{\gamma}^t)\right)\right] \leq \Phi^* + \mathcal{O}(\epsilon). \tag{42}$$

The long-term average expected queue length satisfies

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i,k,n} \mathbb{E}[Q^i_{kn}(t) + q^i_{kn}(t)] = \mathcal{O}\left(\frac{\log^2(\epsilon)}{\sqrt{\epsilon}}\right), \tag{43}$$

where $\mathbf{x}^t(\boldsymbol{\gamma}^t)$ denotes the real-time operations obtained from the Lagrangian minimization (38).

Theorem 3 asserts that by setting $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\epsilon}\log^2(\epsilon))$, Algorithm 2 is asymptotically $\mathcal{O}(\epsilon)$-optimal with an average queue length $\mathcal{O}(\log^2(\epsilon)/\sqrt{\epsilon})$. This implies that the algorithm is able to achieve a near-optimal cost-delay tradeoff $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$; see [31]. Comparing with the standard tradeoff $[\epsilon, 1/\epsilon]$ under Algorithm 1, the learn-and-adapt design of Algorithm 2 remarkably improves the delay performance.

5 Numerical Tests

Numerical tests are provided to validate our analytical claims and demonstrate the merits of the proposed algorithms. Two types of network services are considered on the platform with $N = 50$ VMs. The first type of network

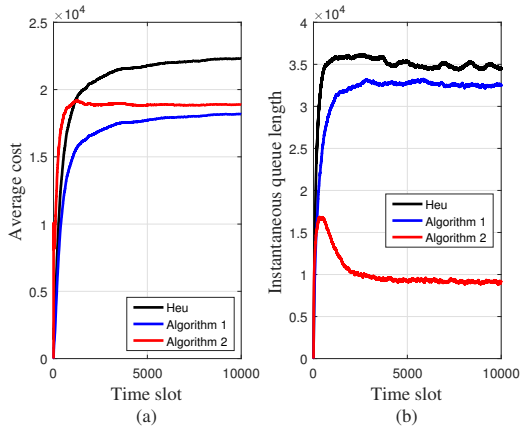


Fig. 3. Comparison of time-average costs and instantaneous queue lengths, where $N = 50$, $\epsilon = 0.1$ and average arrival rate is 220 services/slot.

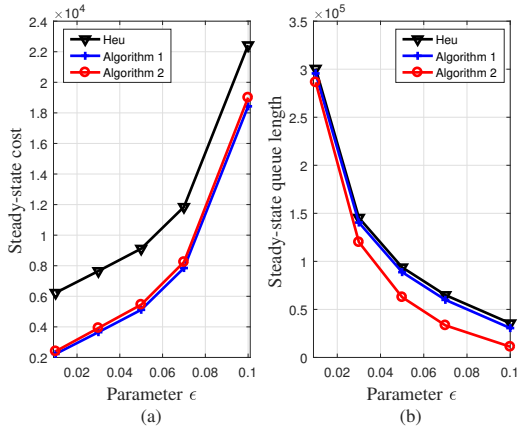


Fig. 4. Comparison of steady-state costs and queue lengths under different ϵ, where $N = 50$ and average arrival rate is 220 services/slot.

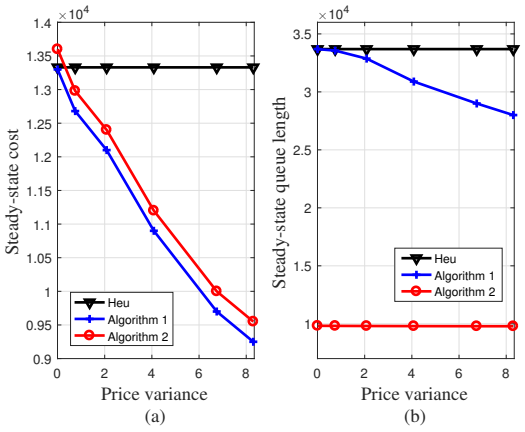


Fig. 5. Comparison of steady-state costs and queue lengths under different price variances, where $N = 50$, $\epsilon = 0.1$ and average arrival rate is 220 services/slot.

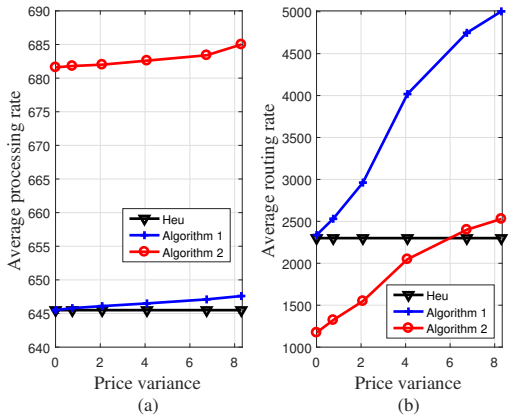


Fig. 6. Comparison of average processing and routing rates for all network services on all VMs under different price variances, where $N = 50$, $\epsilon = 0.1$ and average arrival rate is 220 services/slot.

service is $\{f_1, f_2, f_3\}$ and the second type of network service is $\{f_3, f_1, f_2\}$. Suppose that each service has a size of 1 KB. The processing and routing prices α_n^t and $\beta_{[a,b]}^t$ are uniformly distributed over $[0.1, 1]$ by default; $p_n^{\max}$ and $l_{ab}^{\max}$ are generated from a uniform distribution within $[10, 20]$. The default arrival rate of network services is uniformly distributed with a mean of 220 services/slot. The stepsize is $\eta(t) = 1/\sqrt{t}, \forall t$, the tradeoff variable is $\epsilon = 0.1$, and the bias correction vector is chosen as $\boldsymbol{\theta} = 2\sqrt{\epsilon}\log^2(\epsilon)$. Algorithms are evaluated in a two-timescale scenario, where the placement of VNFs is carried out every $T_\Delta = 5$ slots. In addition to the proposed Algorithms 1 and 2, we also simulate a heuristic algorithm (Heu) as the benchmark, which decides the placement of VNFs and processing/routing rates similarly as Algorithm 1, but with the prices in (23) and (26) replaced by their means. Therefore, the decisions are made only based on queue differences, with no price considerations.

Fig. 3 compares the three algorithms in terms of the time-average cost and the instantaneous queue length. It can be seen from Fig. 3(a) that the time-average cost of Algorithm 2 converges slightly higher than that of Algorithm 1, while the time-average cost of Heu is about 23% larger. Furthermore, Algorithm 2 exhibits faster convergence than Algorithm 1 and Heu, as its time-average cost quickly reaches the optimal steady-state value by leveraging the learning process. Fig. 3(b) shows that Algorithm 2 incurs the shortest queue lengths among the three algorithms, followed by Algorithm 1. Particularly, the aggregated instantaneous queue length of Algorithm 2 is about 72% and 74% smaller than those of Algorithm 1 and Heu, respectively. Clearly, the learn-and-adapt procedure reduces delay without markedly compromising the time-average cost.

Fig. 4 compares the steady-state cost and queue length of the three algorithms, under different stepsize (tradeoff coefficient) ϵ. It is observed that as ϵ grows, the steady-state costs of all three algorithms increase and the steady-state queue lengths declines. This validates our findings in Theorems 1 and 3.

The steady-state cost and queue length are also compared under different price variances in Figs. 5(a) and (b). Here, processing and routing prices are generated with the mean of 0.55 and variance from 3.3×10^{-5} to 8.3×10^{-2}. The costs and queue lengths of Algorithms 1 and 2 decrease as the price variance increases, while those of Heu remain

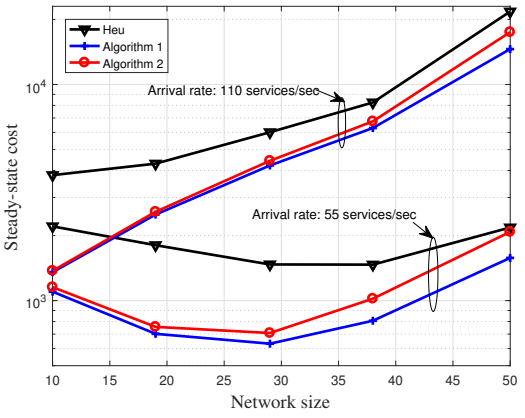


Fig. 7. Comparison of steady-state costs under different network sizes, where $\epsilon = 0.1$.

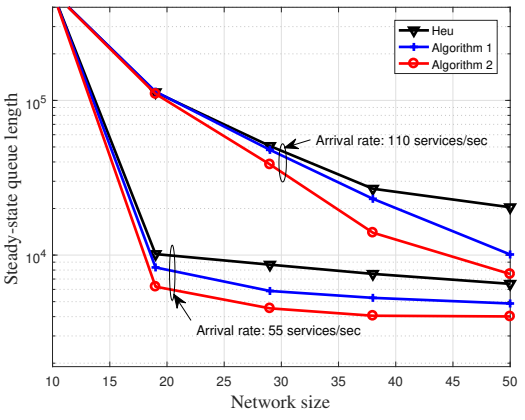


Fig. 8. Comparison of steady-state queue lengths under different network sizes, where $\epsilon = 0.1$.

unchanged. This is because Heu adopts price-independent processing and routing rates, while Algorithms 1 and 2 are able to minimize the cost by taking advantage of price differences among VMs and links.

As further shown in Figs. 6(a) and (b), the average processing and routing rates of Algorithms 1 and 2 rise with the growth of price variance, since the algorithms either choose a lower priced link with a higher routing rate, or a lower priced VM with a higher processing rate.

An interesting finding is that the average backlog of Algorithm 2 is insusceptible to price variances; see Fig. 5(b). This is due to the fact that the algorithm, aiming to reduce the backlog of unfinished network services, achieves the aim by avoiding routing incoming network services to another VMs. This can also be evident from Figs. 6(a) and (b), where VNFs are processed typically at the first encountered corresponding VMs, even at higher processing rates, hence reducing routing rates.

Fig. 7 plots the steady-state costs of Algorithms 1 and 2, and Heu, as the network size N (i.e., the number of VMs) increases. It can be observed that when the arrival rate is 55 services/slot, the costs of Algorithms 1 and 2 first decrease and then increase as the network becomes large. The reduction is because the increased connectivity of VMs helps

TABLE 2
Runtime per timeslot per VM (in milliseconds, c.f., time slot $T = 1$ second/slot)

N	10	20	30	50	75	100
Runtime	0.183	0.222	0.335	0.632	1.06	1.51

increase the routing links and neighboring VMs from which the most cost-effective routing links and neighboring VMs can be chosen for transmitting and processing services. As the network size becomes larger, the number of routing hops of services grows for stabilizing the network, and in turn, increases the costs. We can also see that the costs increase as the average arrival rate of services increases, since more resources are required to accommodate the increased traffic arrivals.

In Fig. 8, we plot the steady-state queue lengths of Algorithms 1 and 2, and Heu, as the network size grows. We can see that Algorithms 1 and 2 are able to reduce the queue length of the network, as compared to Heu. The reductions of Algorithms 1 and 2 are increasingly large, as the arrival rate of services grows. Table 2 shows the runtime performance of Algorithm 1. As analyzed in Section 3.2, Algorithm 1 has a quadratically increasing complexity with regards to the network size N. The approach has a significantly lower complexity and hence higher scalability than the so-called, offline, optimal centralized approach.

6 Conclusions

In this paper, a new distributed online optimization was developed to minimize the time-average cost of NFV, while stabilizing the function queues of VMs. Asymptotically optimal decisions on the placement of VNFs, and the processing and routing of network services were instantly generated at individual VMs, adapting to the topology and stochasticity of the network. A learn-and-adapt approach was further proposed to speed up stabilizing the VMs and achieve a cost-delay tradeoff $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$. Numerical results show that the proposed method is able to reduce the time-average cost of NFV by 23% and reduce the queue length by 74%.

References

[1] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, Dec. 2015.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2016.

[4] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, pp. 1–18, Mar. 2017.

[5] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[6] V. Eramo, M. Ammar, and F. G. Lavacca, "Migration energy aware reconfigurations of virtual network function instances in NFV architectures," *IEEE Access*, vol. 5, pp. 4927–4938, 2017.

[7] R. Riggio, A. Bradai, D. Harutyunyan, and T. Rasheed, "Scheduling wireless virtual networks functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, June 2016.

[8] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1172–1184, Apr. 2016.

[9] W. Chen, I. Paik, and Z. Li, "Cost-aware streaming workflow allocation on geo-distributed data centers," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 256–271, Feb. 2017.

[10] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sept. 2016.

[11] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch, "Cost analysis of initial deployment strategies for virtualized mobile core network functions," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 60–66, Dec. 2015.

[12] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications*, 2015, pp. 1346–1354.

[13] V. Eramo, A. Tosti, and E. Miucci, "Server resource dimensioning and routing of service function chain in NFV network architectures," *J. Electr. Comput. Eng.*, vol. 2016, no. 9, pp. 1–12, 2016.

[14] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, London, UK, 13–17 Apr. 2015.

[15] R. Wen, G. Feng, W. Tan, R. Ni, S. Qin, and G. Wang, "Protocol function block mapping of software defined protocol for 5G mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 7, pp. 1651–1665, 2018.

[16] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, and G. B. Giannakis, "Real-time energy trading and future planning for fifth-generation wireless communications," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 24–30, Aug. 2017.

[17] X. Wang, Y. Zhang, T. Chen, and G. B. Giannakis, "Dynamic energy management for smart-grid-powered coordinated multipoint systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1348–1359, May 2016.

[18] X. Wang, X. Chen, T. Chen, L. Huang, and G. B. Giannakis, "Two-scale stochastic control for integrated multipoint communication systems with renewables," *IEEE Trans. Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.

[19] X. Wang, T. Chen, X. Chen, X. Zhou, and G. B. Giannakis, "Dynamic resource allocation for smart-grid powered MIMO downlink transmissions," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3354–3365, Dec. 2016.

[20] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, and M. J. Neely, "Power cost reduction in distributed data centers: A two-time-scale approach for delay tolerant workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 200–211, Jan. 2013.

[21] S. Sun, M. Dong, and B. Liang, "Distributed real-time power balancing in renewable-integrated power grids with storage and flexible loads," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2337–2349, Sept. 2016.

[22] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined network," *IEEE J. Sel. Areas Commun.*, accepted in Aug. 2018.

[23] M. J. Neely, "Optimal backpressure routing for wireless networks with multi-receiver diversity," *Ad Hoc Networks*, vol. 7, no. 5, pp. 862–881, 2009.

[24] L. Huang and M. J. Neely, "The optimality of two prices: Maximizing revenue in a stochastic communication system," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 406–419, Apr. 2010.

[25] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, June 2012.

[26] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, R. P. Liu, and G. B. Giannakis, "Distributed stochastic optimization of network function virtualization," in *Proc. IEEE GLOBECOM*, Singapore, Dec. 2017.

[27] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz, and H. Ahmed, "Understanding and applying precision time protocol," in *Smart Grid*, 2016, pp. 1–7.

[28] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *IEEE INFOCOM*, 2017, pp. 1–9.

[29] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[30] Q. Zhu and R. Boutaba, "Nonlinear quadratic pricing for concavifiable utilities in network rate control," in *Proc. IEEE GLOBECOM*, New Orleans, LA, Dec. 2008.

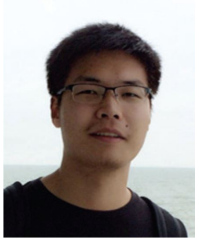
[31] T. Chen, Q. Ling, and G. B. Giannakis, "Learn-and-adapt stochastic dual gradients for network resource allocation," *IEEE Trans. Contr. Netw. Syst.*, Available online: https://arxiv.org/pdf/1703.01673v1.pdf, 2017.

[32] D. M. Himmelblau, *Applied nonlinear programming*. McGraw-Hill Companies, 1972.

Xiaojing Chen (S'14) received the B.E. degree in Communication Science and Engineering from Fudan University, China in 2013. Currently she is working toward her Ph.D. degrees in both Fudan University and Macquarie University. Her research interests include wireless communications, energy-efficient communications, stochastic network optimization and network functions virtualization. She received a National Scholarship from China in 2016.

Wei Ni (SM'15) received the B.E. and Ph.D. degrees in Electronic Engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. Currently he is a Senior Scientist, and Team and Project Leader at CSIRO, Australia. He also holds honorary positions at the University of New South Wales (UNSW), Macquarie University (MQ) and the University of Technology Sydney (UTS). Prior to this he was a post-doctoral research fellow at Shanghai Jiaotong University (2005-2008), Research Scientist and Deputy Project Manager at the Bell Labs R&I Center, Alcatel/Alcatel-Lucent (2005-2008), and Senior Researcher at Devices R&D, Nokia (2008-2009). His research interests include optimization, game theory, graph theory, as well as their applications to network and security.

Dr Ni serves as Editor for Hindawi Journal of Engineering since 2012, secretary of IEEE NSW VTS Chapter since 2015, Track Chair for VTC-Spring 2016 and 2017, and Publication Chair for BodyNet 2015. He also served as Student Travel Grant Chair for WPMC 2014, Program Committee Member of CHINACOM 2014, TPC member of IEEE ICC'14, ICCC'15, EICE'14, and WCNC'10.



Tianyi Chen (S'14) received the B. Eng. degree (with highest honors) in Communication Science and Engineering from Fudan University, and the M.Sc. degree in Electrical and Computer Engineering (ECE)from the University of Minnesota (UMN), in 2014 and 2016, respectively. Since July 2016, he has been working toward his Ph.D. degree at UMN. His research interests lie in online learning, online convex optimization, and stochastic network optimization with applications to smart grids, and Internet-of-Things. He was in the Best Student Paper Award finalist of the Asilomar Conference on Signals, Systems, and Computers. He received the National Scholarship from China in 2013, UMN ECE Department Fellowship in 2014, and the UMN Doctoral Dissertation Fellowship in 2017.

Iain B. Collings (Fellow'15) received the B.E. degree in electrical and electronic engineering from the University of Melbourne in 1992, and the Ph.D. degree in systems engineering from the Australian National University in 1995. Prior to his current position as Head of Department and Professor of Engineering at Macquarie University, he was Research Director and OCE Science Leader of theWireless and Networking Technologies Laboratory at the CSIRO ICT Centre, an Associate Professor at the University of Sydney (1999C2005), a Lecturer at the University of Melbourne (1996C1999), and a Research Fellow in the Australian Cooperative Research Centre for Sensor Signal and Information Processing (1995).

He was elected as a 2015 IEEE Fellow. He has published over 300 research papers in the area of wireless digital communications. In 2009, he was awarded the Engineers Australia IREE Neville Thiele Award for outstanding achievements in engineering, and in 2011 he was a recipient of the IEEE CommSoc Stephen O. Rice Best Paper Award for IEEE TRANSACTIONS ON COMMUNICATIONS. Dr. Collings served as an Editor for the IEEE TRANSACTIONS ONWIRELESS COMMUNICATIONS (2002C2009), and the Physical Communication Journal (2008C2012). He has served as a Co/Vice-Chair of the Conference Technical Program Committees for IEEE International Conference on Communications (ICC) 2013, IEEE Vehicular Technology Conference (VTC) Spring 2011, IEEE Wireless Communications and Networking Conference (WCNC) 2010, and IEEE VTC Spring 2006. He is a founding organizer of the Australian Communication Theory Workshops 2000C2013. He also served as the Chair of the Joint Communications & Signal Processing Chapter in the IEEE NSW Section (2008C2010), and as Secretary of the IEEE NSW Section (2010).

Xin Wang (SM'09) received the B.Sc. degree and the M.Sc. degree from Fudan University, Shanghai, China, in 1997 and 2000, respectively, and the Ph.D. degree from Auburn University, Auburn, AL, in 2004, all in electrical engineering.

From September 2004 to August 2006, he was a Postdoctoral Research Associate with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis. In August 2006, he joined the Department of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, as an Assistant Professor, and then an Associate Professor from August 2010. He is now a Professor with the Department of Communication Science and Engineering, Fudan University, China. His research interests include stochastic network optimization, energy-efficient communications, cross-layer design, and signal processing for communications. He is an Associate Editor for the IEEE Transactions on Signal Processing, and an Editor for the IEEE Transactions on Vehicular Technology. He was an Associate Editor for the IEEE Signal Processing Letters.

Ren Ping Liu (SM'14) is a Professor at the School of Electrical and Data Engineering in University of Technology Sydney, where he leads the Network Security Lab in the Global Big Data Technologies Centre. He is also the Research Program Leader of the Digital Agrifood Technologies in Food Agility CRC, a government/research/industry initiative to empower Australia's food industry through digital transformation. Prior to that he was a Principal Scientist at CSIRO, where he led wireless networking research activities. He specialises in protocol design and modelling, and has delivered networking solutions to a number of government agencies and industry customers. Professor Liu was the winner of Australian Engineering Innovation Award and CSIRO Chairman medal. His research interests include Markov analysis and QoS scheduling in WLAN, VANET, IoT, LTE, 5G, SDN, and network security. Professor Liu has over 100 research publications, and has supervised over 30 PhD students.

Professor Liu is the founding chair of IEEE NSW VTS Chapter and a Senior Member of IEEE. He served as Technical Program Committee chair and Organising Committee chair in a number of IEEE Conferences. Ren Ping Liu received his B.E.(Hon) and M.E. degrees from Beijing University of Posts and Telecommunications, China, and the Ph.D. degree from the University of Newcastle, Australia.

Georgios. B. Giannakis (Fellow'97) received his Diploma in Electrical Engr. from the Ntl. Tech. Univ. of Athens, Greece, 1981. From 1982 to 1986 he was with the Univ. of Southern California (USC), where he received his MSc. in Electrical Engineering, 1983, MSc. in Mathematics, 1986, and Ph.D. in Electrical Engr., 1986. He was with the University of Virginia from 1987 to 1998, and since 1999 he has been a professor with the Univ. of Minnesota, where he holds an Endowed Chair in Wireless Telecommunications, a University of Minnesota McKnight Presidential Chair in ECE, and serves as director of the Digital Technology Center.

His general interests span the areas of communications, networking and statistical signal processing - subjects on which he has published more than 400 journal papers, 700 conference papers, 25 book chapters, two edited books and two research monographs (h-index 124). Current research focuses on learning from Big Data, wireless cognitive radios, and network science with applications to social, brain, and power networks with renewables. He is the (co-) inventor of 30 patents issued, and the (co-) recipient of 8 best paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SP Society.