Distributed Stochastic Optimization of Network Function Virtualization

Xiaojing Chen[1,4], Wei Ni[2], Tianyi Chen[3], Iain B. Collings[4], Xin Wang[1], Ren Ping Liu[5], and Georgios B. Giannakis[3]
[1]Key Lab of EMW Information (MoE), Dept. of Commun. Sci. & Engr., Fudan University, China
[2]Digital Productivity and Service (DP&S) Flagship, CSIRO, Australia
[3]Dept. of Elec. & Comput. Engr. and Digital Technology Center, University of Minnesota, USA
[4]Dept. of Engr., Macquarie University, Australia
[5]School of Comput. & Commun., University of Technology Sydney, Australia

***Abstract*—Decoupling network services from underlying hardware, network function virtualization (NFV) is expected to significantly improve agility and reduce network cost. However, network services, sequences of network functions, need to be processed in specific orders at specific types of virtual machines (VMs), which couples decisions of VMs on processing or routing network services. Built on a new stochastic dual gradient method, our approach suppresses the couplings, minimizes the time-average cost of NFV, stabilizes queues at VMs, and reduces the backlogs of unprocessed services through online learning and adaptation. Asymptotically optimal decisions are instantly generated at individual VMs, with 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 30% and reduce the queue length (or delay) by 83%, as compared to existing non-stochastic approaches.**

***Index Terms*—Network function virtualization, virtual machine, distributed optimization, stochastic approximation.**

I. Introduction

Decoupling dedicated hardware from network services and replacing with programmable virtual machines (VMs), Network Function Virtualization (NFV) is able to provide critical network functions on top of optimally shared physical infrastructure [1]. This can avoid disproportional hardware investments on short-lived functions, and adapt quickly as network functions evolve [2].

A network service (or service chain) can consist of multiple virtual network functions (VNFs), which need to be run in a predefined order at different VMs running different software and processes [3]. Challenges arise from optimal decision-makings of processing or routing VNFs at each VM, especially in large-scale network platforms. On one hand, given the sequence of VNFs 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 [4]. Prices can also vary for the service of a VM, depending on the pricing policy of the service providers. The stochasticity results in couplings of optimal decisions in time.

Work in this paper was supported by the National Natural Science Foundation of China grant 61671154, the Innovation Program of Shanghai Municipal Education Commission; and US NSF 1509005, 1508993, 1423316, 1442686, 1202135.

These are open challenges and have not been captured in previous works. Earlier works were focused on the placement of VNFs under the assumption of persistent arrivals of VMs [5] or full knowledge on the statistics of the arrivals [6], where VNFs were instantly processed at the VMs admitting them and routing was overlooked. Taking network service chains into account, recent works studied optimal decision-makings on processing and routing VNFs, under the assumption of persistent service arrivals [7]. NP-complete mixed integer linear programming (MILP) was formulated to minimize the delay of network service chains [8]. These heuristics still need to run in a centralized manner, limiting scalability. Moreover, none of these works have taken random network service arrivals or dynamic pricing into account.

In this paper, we propose a new distributed online optimization of NFV, where asymptotically optimal decisions of processing or routing VNFs are instantly generated at individual VMs, adapting to the topology and stochasticity of the network. Capturing random service arrivals and time-varying prices, a new stochastic dual gradient method is developed to decouple optimal decision-makings across different VMs and different time slots, and minimize 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 offline-generated, causality-violating, global optimum by tuning the coefficient ϵ.

Another important contribution of this paper is that we further speed up stabilizing the VMs and improve the cost-delay tradeoff to $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$. A learn-and-adapt approach is designed to accelerate the ascent of the aforementioned gradients and reduce the steady-state queue lengths by learning the statistics of the gradients from history.

The rest of the paper is organized as follows. In Section II, the system model is described. In Section III, the distributed online optimization of service processing and routing is developed. The learn-and-adapt enhancement is proposed in Section IV. Numerical tests are provided in Section V, followed by concluding remarks in Section VI.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

II. System Model

Consider a platform consisting of N VMs, supporting K VNFs, and operating in a (possibly infinite) scheduling horizon consisting of T slots with a normalized slot duration "1". Assume that every VM can admit network services to be processed on the platform, and output the results. $\mathcal{N} = \{1, \ldots, N\}$ collects the N VMs. Let 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 the software for a single VNF. $\mathcal{N}$ can be divided into K subgroups, denoted by $\mathcal{N}_k$, $k = 1, \ldots, K$. $\mathcal{N}_k$ consists of the VMs which can process VNF f_k.

Let $\mathcal{I}$ collect all possible types of network services, each of which is a permuted sequence of $\{f_1, \ldots, f_K\}$. In this sense, a network service needs to traverse among multiple subgroups of VMs, until all the VNFs within the service are processed in the correct order. Consider network services of type i and VM n. U_n^i denotes the set of the corresponding upstream VMs, D_n^i the set of the downstream VMs, and S_n^i the set of VMs that are able to process the same type of VNF as, and have virtual links to, VM n. U_n^i and D_n^i depend on the order of VNFs to be processed in the network service.

We design up to $2|\mathcal{I}|$ First-In-First-Out (FIFO) queues at each VM n. Half of the queues buffer network services with f_k being the head-of-line (HOL) unprocessed VNF; and the other half buffer the result of the first half with f_k processed and to be routed to downstream VMs for further processing. Let $Q_n^i(t)$ and $q_n^i(t)$ denote the queue lengths of the unprocessed and processed network services of type i at node n, slot t. Let $\mathbf{Q}(t) = \{Q_n^i(t), \forall n \in \mathcal{N}, i \in \mathcal{I}\}$, $\mathbf{q}(t) = \{q_n^i(t), \forall n \in \mathcal{N}, i \in \mathcal{I}\}$, and $\mathbf{A}(t) = \{\mathbf{Q}(t), \mathbf{q}(t)\}$. $R_n^{i,t} \leq R^{\max}$ denotes the arrival rate (in services per slot) of a new network service of type i at node n, where $R^{\max}$ is the maximum arrival rate.

We assume that any VM n or directional link $[a, b]$ can only process or transmit a single network service per slot. With the data rate (in service per slot) of network services of type i over directional link $[a, b]$, denoted by $u_{[a,b]}^i(t)$, it is easy to see that at any time t, we have

$$u_{[a,b]}^i(t) \geq 0, \ \sum_i u_{[a,b]}^i(t) = u_{[a,b]}^{i^*}(t) \leq u_{[a,b]}^{\max}, \ \forall i, [a, b] \quad (1)$$

where $u_{[a,b]}^{\max}$ is the maximum data rate over link $[a, b]$, and a network service of type i^* is selected to be forwarded.

Let $\delta_n^i(t)$ denote the processing rate (in services per slot) for network services of type i on VM n. We also have

$$\delta_n^i(t) \geq 0, \ \sum_i \delta_n^i(t) = \delta_n^{i^*}(t) \leq \delta_n^{\max}, \ \forall n, \quad (2)$$

where $\delta_n^{\max}$ is the maximum processing rate of VM n, and a network service of type i^* is selected to be processed.

Therefore, the queue length of unprocessed network services of type i at VM n follows:

$$\begin{aligned} Q_n^i(t+1) &= Q_n^i(t) - \sum_{b \in S_n^i} u_{[n,b]}^i(t) - \delta_n^i(t) \\ &+ \sum_{a \in U_n^i} u_{[a,n]}^i(t) + \sum_{c \in S_n^i} u_{[c,n]}^i(t) + R_n^{i,t}, \quad \forall i, t, n. \end{aligned} \quad (3)$$

The queue length of processed network services of type i at VM n follows:

$$q_n^i(t+1) = q_n^i(t) - \sum_{d \in D_n^i} u_{[n,d]}^i(t) + \delta_n^i(t), \quad \forall i, t, n, \quad (4)$$

where $q_n^i(t) = 0$ in the case that the last VNF of network services of type i is processed at VM n and therefore output from the platform.

We can define the network is *stable* if and only if the following is met [9]:

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

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

$$\Phi(u_{[a,b]}^i(t), \delta_n^i(t)) := \sum_{a,b,i} f(u_{[a,b]}^i(t)) + \sum_{n,i} h(\delta_n^i(t)), \quad (6)$$

where $f(u_{[a,b]}^i(t)) = \beta_{[a,b]}^t (u_{[a,b]}^i(t))^2$ and $h(\delta_n^i(t)) = \alpha_n^t (\delta_n^i(t))^2$ are the prices that the network service provider charges over usages of links and VMs, respectively, following a quadratic pricing policy [10]. $\beta_{[a,b]}^t$ is the time-varying price for delivering services over link $[a, b]$ and α_n^t is the time-varying price for processing services at VM n.

III. Distributed Online Optimization of Processing and Routing

The objective of this paper is to minimize the time-average cost of NFV on a network platform while preserving the stability of the platform (i.e., finite queue lengths), under random network service arrivals and prices. This is to be achieved by making stochastically optimal decisions on processing or routing network services at every VM and time slot in a distributed fashion. Let $\mathbf{x}^t := \{u_{[a,b]}^i(t), \forall [a, b], i; \delta_n^i(t), \forall n, i\}$ and $\mathcal{X} := \{\mathbf{x}^t, \forall t\}$. The problem of interest is to find

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

where the expectation of $\Phi(\mathbf{x}^t)$ is taken over all randomnesses. The service arrival rate $\{R_n^{i,t}, \forall n, i, t\}$ and the routing and processing prices $\{\beta_{[a,b]}^t, \alpha_n^t, \forall [a, b], n, t\}$ are all random.

A. Dual gradient and asymptotic optimality

It is difficult to solve (7) since 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 [9]

$$\begin{aligned} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[&\sum_{a \in U_n^i} u_{[a,n]}^i(t) + \sum_{c \in S_n^i} u_{[c,n]}^i(t) + R_n^{i,t} \\ &- \sum_{b \in S_n^i} u_{[n,b]}^i(t) - \delta_n^i(t)] \leq 0, \quad \forall i, n. \end{aligned} \quad (8a)$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\delta_n^i(t)-\sum_{d\in D_n^i}u_{[n,d]}^i(t)]\le 0,\quad \forall i,n. \tag{8b}$$

As a result, (7) can be relaxed as

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

Compared to (7), (9) eliminates the time coupling among variables $\{\mathbf{A}(t),\forall t\}$ by replacing (3), (4) and (5) with (8). (9) is a relaxation of (7) with its optimal objective $\tilde{\Phi}^*\le\Phi^*$.

We can take a stochastic gradient approach to solving (9) in an asymptotically optimal manner. Concatenate the random parameters into a state vector $\boldsymbol{s}^t:=[R_n^{i,t},\beta_{[a,b]}^t,\alpha_n^t,\forall[a,b],n,i]$. Suppose that $\boldsymbol{s}^t$ is independent and identically distributed (i.i.d.) across time slots, then we can rewrite (9) as [9]

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

$$\text{s.t.}\quad 0\le\sum_i u_{[a,b]}^i(\boldsymbol{s}^t)=u_{[a,b]}(\boldsymbol{s}^t)\le u_{\max}, \tag{10b}$$

$$0\le\sum_i\delta_n^i(\boldsymbol{s}^t)=\delta_n(\boldsymbol{s}^t)\le\delta_{\max},\ \forall n \tag{10c}$$

$$\mathbb{E}[\sum_{a\in U_n^i}u_{[a,n]}^i(\boldsymbol{s}^t)+\sum_{c\in S_n^i}u_{[c,n]}^i(\boldsymbol{s}^t)+R_n^{i,t}-\sum_{b\in S_n^i}u_{[n,b]}^i(\boldsymbol{s}^t)-\delta_n^i(\boldsymbol{s}^t)]\le 0 \tag{10d}$$

$$\mathbb{E}[\delta_n^i(\boldsymbol{s}^t)-\sum_{d\in D_n^i}u_{[n,d]}^i(\boldsymbol{s}^t)]\le 0, \tag{10e}$$

where $u_{[a,b]}^i(\boldsymbol{s}^t):=u_{[a,b]}^i(t),\delta_n^i(\boldsymbol{s}^t):=\delta_n^i(t),\forall[a,b],n,i$, and $\Phi(\mathcal{X}(\boldsymbol{s}^t);\boldsymbol{s}^t):=\Phi(\mathbf{x}^t)$.

Let $\mathcal{X}^t$ denote the set of $\{u_{[a,b]}^i(t),\forall[a,b],i;\delta_n^i(t),\forall n,i\}$ satisfying constraints (1) and (2) per t. $\lambda_{n,1}^i$ and $\lambda_{n,2}^i$ denote the Lagrange multipliers associated with the constraints (10d) and (10e). With a convenient notation $\boldsymbol{\lambda}:=\{\lambda_{n,1}^i,\lambda_{n,2}^i,\forall n,i\}$, the partial Lagrangian function of (10) is

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

where the instantaneous Lagrangian is given by

$$\begin{aligned}L^t(\mathbf{x}^t,\boldsymbol{\lambda}):=\Phi(\mathbf{x}^t)&+\sum_{i,n}\lambda_{n,1}^i(t)(\sum_{a\in U_n^i}u_{[a,n]}^i(t)\\&+\sum_{c\in S_n^i}u_{[c,n]}^i(t)+R_n^{i,t}-\sum_{b\in S_n^i}u_{[n,b]}^i(t)-\delta_n^i(t))\\&+\sum_{i,n}\lambda_{n,2}^i(t)(\delta_n^i(t)-\sum_{d\in D_n^i}u_{[n,d]}^i(t)).\end{aligned} \tag{12}$$

Notice that the instantaneous objective $\Phi(\mathbf{x}^t)$ and the instantaneous constraints associated with $\boldsymbol{\lambda}$ are parameterized by the observed state $\boldsymbol{s}^t$ at time t.

As a result, the Lagrange dual function is given by

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

and the dual problem of (9) is: $\max_{\boldsymbol{\lambda}\succeq 0}\ D(\boldsymbol{\lambda})$.

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

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

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

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

$$\begin{aligned}g_{\lambda_{n,1}^i}(t)=\mathbb{E}[&\sum_{a\in U_n^i}u_{[a,n]}^i(t)+\sum_{c\in S_n^i}u_{[c,n]}^i(t)+R_n^{i,t}\\&-\sum_{b\in S_n^i}u_{[n,b]}^i(t)-\delta_n^i(t)]\end{aligned} \tag{15a}$$

$$g_{\lambda_{n,2}^i}(t)=\mathbb{E}[\delta_n^i(t)-\sum_{d\in D_n^i}u_{[n,d]}^i(t)] \tag{15b}$$

where $\mathbf{x}^t:=\{u_{[a,b]}^i(t),\forall[a,b],i;\delta_n^i(t),\forall n,i\}$ are given by

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

Note that an impassable challenge associated with (15) is sequentially taking expectations over the random vector $\boldsymbol{s}^t$ to compute $\boldsymbol{g}(t)$. This would require high-dimensional integration over an unknown probabilistic distribution function of $\boldsymbol{s}^t$. This requirement is impractical since the associated computational complexity could be prohibitively high.

To bypass this impasse, we propose to rely on a stochastic dual gradient approach. Specifically, dropping $\mathbb{E}$ from (15), we propose the following iterations

$$\begin{aligned}\tilde{\lambda}_{n,1}^i(t+1)=\tilde{\lambda}_{n,1}^i(t)+\epsilon[&\sum_{a\in U_n^i}u_{[a,n]}^i(t)+\sum_{c\in S_n^i}u_{[c,n]}^i(t)\\&+R_n^{i,t}-\sum_{b\in S_n^i}u_{[n,b]}^i(t)-\delta_n^i(t)]^+\end{aligned} \tag{17a}$$

$$\tilde{\lambda}_{n,2}^i(t+1)=\tilde{\lambda}_{n,2}^i(t)+\epsilon[\delta_n^i(t)-\sum_{d\in D_n^i}u_{[n,d]}^i(t)]^+ \tag{17b}$$

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

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 is made possible due to the decoupling of optimization variables over time in (9).

Relying on the so-called Lyapunov optimization technique in [9], [11], [12], we can formally establish that:

Proposition 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

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

where 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 random realizations.

The proofs for all propositions and theorems are omitted due to limited space, and can be found in the extended journal version [13].

Proposition 2: Assume that there exists a stationary policy $\mathcal{X}$ and $\mathbb{E}[\sum_{a\in U_n^i} u_{[a,n]}^i(t) + \sum_{c\in S_n^i} u_{[c,n]}^i(t) + R_n^{i,t} - \sum_{b\in S_n^i} u_{[n,b]}^i(t) - \delta_n^i(t)] \leq -\zeta$, and $\mathbb{E}[\delta_n^i(t) - \sum_{d\in D_n^i} u_{[n,d]}^i(t)] \leq -\zeta$, where $\zeta > 0$ is a slack vector constant, then all queues are stable, and the time-average queue length satisfies:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{n,i} \mathbb{E}[Q_n^i(t) + q_n^i(t)] = \mathcal{O}(\frac{1}{\epsilon}). \tag{18}$$

Propositions 1 and 2 assert that the time-average cost of (10) obtained by the stochastic dual gradient approach converges to a region within an optimality gap of $\mathcal{O}(\epsilon)$, which vanishes as the stepsize $\epsilon \to 0$. The typical tradeoff from the stochastic network optimization holds in this case [9]: an $\mathcal{O}(1/\epsilon)$ queue length is necessary, when an $\mathcal{O}(\epsilon)$ close-to-optimal cost is achieved.

B. Distributed online implementation

The dual iteration (17) coincides with (3) and (4) for $\hat{\lambda}_{n,1}^i(t)/\epsilon = Q_n^i(t)$ and $\hat{\lambda}_{n,2}^i(t)/\epsilon = q_n^i(t), \forall n, i, t$; this can be interpreted by using the concept of virtual queue of this parallelism [9]. With $\hat{\lambda}_{n,1}^i(t)$ substituted by $\epsilon Q_n^i(t)$ and $\hat{\lambda}_{n,2}^i(t)$ substituted by $\epsilon q_n^i(t)$, we can obtain the desired $\mathbf{x}^t(\mathbf{A}(t))$ by solving the following problem:

$$\begin{aligned}\min_{\mathbf{x}^t\in\mathcal{X}^t} \frac{1}{\epsilon}\Phi(\mathbf{x}^t) + \sum_{n,i} Q_n^i(t)[\sum_{a\in U_n^i} u_{[a,n]}^i(t) + \sum_{c\in S_n^i} u_{[c,n]}^i(t) \\ + R_n^{i,t} - \sum_{b\in S_n^i} u_{[n,b]}^i(t) - \delta_n^i(t)] \\ + \sum_{n,i} q_n^i(t)[\delta_n^i(t) - \sum_{d\in D_n^i} u_{[n,d]}^i(t)].\end{aligned} \tag{19}$$

Through rearrangement, (19) is equivalent to

$$\begin{aligned}&\min_{\mathbf{x}^t\in\mathcal{X}^t} \sum_{n,i}[\frac{\alpha_n^t}{\epsilon}(\delta_n^i(t))^2 - (Q_n^i(t) - q_n^i(t))\delta_n^i(t)] \\ &+ \sum_{n,i,b\in S_n^i}[\frac{\beta_{[n,b]}^t}{\epsilon}(u_{[n,b]}^i(t))^2 - (Q_n^i(t) - Q_b^i(t))u_{[n,b]}^i(t)] \\ &+ \sum_{n,i,d\in D_n^i}[\frac{\beta_{[n,d]}^t}{\epsilon}(u_{[n,d]}^i(t))^2 - (q_n^i(t) - Q_d^i(t))u_{[n,d]}^i(t)],\end{aligned} \tag{20}$$

which can be readily solved by decoupling between $\delta_n^i(t)$, $u_{[n,b]}^i(t)$ and $u_{[n,d]}^i(t)$ and between the VMs, and evaluating the first-order derivatives of the decoupled objectives. The optimal solutions for network services of type i at VM n are ${\delta_n^i}^*(t) = \max\{\frac{\epsilon(Q_n^i(t)-q_n^i(t))}{2\alpha_n^t}, 0\}$, ${u_{[n,b]}^i}^*(t) = \max\{\frac{\epsilon(Q_n^i(t)-Q_b^i(t))}{2\beta_{[n,b]}^t}, 0\}$, and ${u_{[n,d]}^i}^*(t) = \max\{\frac{\epsilon(q_n^i(t)-Q_d^i(t))}{2\beta_{[n,d]}^t}, 0\}$, with the corresponding objectives:

$$W_n^i := \begin{cases} -\frac{\epsilon(Q_n^i(t)-q_n^i(t))^2}{4\alpha_n^t}, & \text{if } Q_n^i(t) - q_n^i(t) > 0 \\ 0, & \text{if } Q_n^i(t) - q_n^i(t) \leq 0 \end{cases}$$

$$W_{[n,b]}^i := \begin{cases} -\frac{\epsilon(Q_n^i(t)-Q_b^i(t))^2}{4\beta_{[n,b]}^t}, & \text{if } Q_n^i(t) - Q_b^i(t) > 0 \\ 0, & \text{if } Q_n^i(t) - Q_b^i(t) \leq 0 \end{cases}$$

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: Repeatedly send network services to the VM processor or outgoing links with the minimum non-zero queue-price weights, until either the processor and all outgoing links are scheduled or the remaining weights are all zero.
4: Update $Q_n^i(t)$ and $q_n^i(t)$ for all nodes and services via the dynamics (3) and (4).
5: **end for**

$$W_{[n,d]}^i := \begin{cases} -\frac{\epsilon(q_n^i(t)-Q_d^i(t))^2}{4\beta_{[n,d]}^t}, & \text{if } q_n^i(t) - Q_d^i(t) > 0 \\ 0, & \text{if } q_n^i(t) - Q_d^i(t) \leq 0 \end{cases}$$

$$\forall i, b \in S_n^i, d \in D_n^i. \tag{21}$$

In light of this, at each slot, a VM can prioritize the queues of different types of networks services, and process or route services from the queue with the highest priority. The priority is ranked based on (21). For this reason, we refer to W_n^i, $W_{[n,b]}^i$ and $W_{[n,d]}^i$ as queue-price weights, and 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 the selected non-zero weights, as summarized in Algorithm 1.

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 weights, comply with (17) and therefore preserve the asymptotic optimality of the entire network, as dictated in Propositions 1 and 2.

Also note that the actual routing decisions are discretized, since the number of network services delivered over a link is integer for the integrity of the services. To this end, the decisions $\mathbf{x}^t$ needs to be replaced by $\lceil \mathbf{x}^t \rceil$. It can be proved that the discretization does not violate the asymptotic optimality of Algorithm 1 [13].

IV. Distributed Online Learn-and-Adapt Optimization

With a cost-delay tradeoff $[\epsilon, 1/\epsilon]$, Algorithm 1 may accumulate long queues per VM to achieve the near-optimality (with sufficiently small ϵ). We propose to improve the tradeoff and reduce the queue lengths through learning and adaptation [10]. By incrementally learn the Lagrangian multipliers from observed data, we can speed up the convergence of the multipliers driven by the learning process.

In the proposed learn-and-adapt scheme, with the online learning of $\tilde{\lambda}_{n,1}^i(t)$ and $\tilde{\lambda}_{n,2}^i(t), \forall n, i$ 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

Algorithm 2 Distributed Online Learn-and-Adapt Optimization of NFV

1: **for** $t = 1, 2 \dots$ **do**
2: **Online processing and routing (1st gradient):**
3: Construct the effective dual variable via (22b), observe the current state $\boldsymbol{s}^t$, and obtain processing and routing decisions $\mathbf{x}^t(\boldsymbol{\gamma}^t)$ by minimizing online Lagrangian (22a).
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 (23b).
7: Update the empirical dual variable $\hat{\boldsymbol{\lambda}}^{t+1}$ via (23a).
8: **end for**

routing network services, as given by [cf. (16)]

$$\mathbf{x}^t(\boldsymbol{\gamma}^t) = \arg \min_{\mathbf{x}^t \in \mathcal{X}^t} L^t(\mathbf{x}^t, \boldsymbol{\gamma}^t) \tag{22a}$$

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

$$\underbrace{\boldsymbol{\gamma}^t}_{\text{effective multiplier}} = \underbrace{\hat{\boldsymbol{\lambda}}^t}_{\text{statistical learning}} + \underbrace{\epsilon \mathbf{A}(t) - \boldsymbol{\theta}}_{\text{online adaptation}}, \tag{22b}$$

where $\hat{\boldsymbol{\lambda}}^t := \{\hat{\lambda}^i_{n,1}(t), \hat{\lambda}^i_{n,2}(t), \forall n, i\}$ 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 1.

For a better illustration of the effective multiplier in (22b), 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 $\epsilon \mathbf{A}(t)$ 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

$$\begin{aligned}
\hat{\lambda}^i_{n,1}(t+1) &= \hat{\lambda}^i_{n,1}(t) + \eta(t)[\sum_{a \in U^i_n} u^i_{[a,n]}(\hat{\lambda}^i_{n,1}(t)) + R^{i,t}_n \\
&+ \sum_{c \in S^i_n} u^i_{[c,n]}(\hat{\lambda}^i_{n,1}(t)) - \sum_{b \in S^i_n} u^i_{[n,b]}(\hat{\lambda}^i_{n,1}(t)) - \delta^i_n(\hat{\lambda}^i_{n,1}(t))]^+ \\
\hat{\lambda}^i_{n,2}(t+1) &= \hat{\lambda}^i_{n,2}(t) + \eta(t)[\delta^i_n(\hat{\lambda}^i_{n,2}(t)) - \sum_{d \in D^i_n} u^i_{[n,d]}(\hat{\lambda}^i_{n,2}(t))]^+
\end{aligned} \tag{23a}$$

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

$$\mathbf{x}^t(\hat{\boldsymbol{\lambda}}^t) = \arg \min_{\mathbf{x}^t \in \mathcal{X}^t} L^t(\mathbf{x}^t, \hat{\boldsymbol{\lambda}}^t). \tag{23b}$$

With learn-and-adaption incorporated, Algorithm 2 takes an additional learning step in Algorithm 1, i.e., (23a), 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.

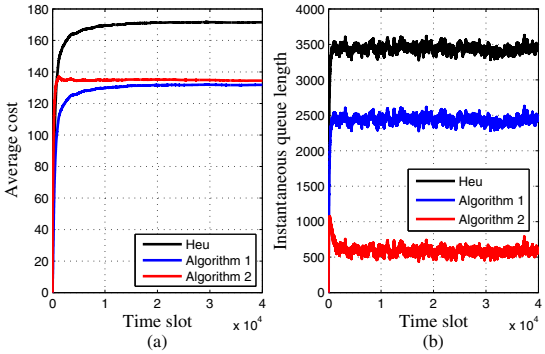

Fig. 1. Comparison of time-average costs and instantaneous queue lengths, where $\epsilon = 0.1$.

Using the Learn-and-adapt approach, we are ready to arrive at the following theorem [10, Theorems 2-3].

Theorem 1: Suppose that the assumptions in Propositions 1 and 2 are satisfied. Then with $\boldsymbol{\gamma}^t$ defined in (22b) and $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\epsilon}\log^2(\epsilon))$, Algorithm 2 yields a near-optimal solution for (7) in the sense that

$$\Phi^* \leq \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\Phi\left(\mathbf{x}^t(\boldsymbol{\gamma}^t)\right)\right] \leq \Phi^* + \mathcal{O}(\epsilon). \tag{24}$$

The long-term average expected queue length satisfies

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{n,i} \mathbb{E}[Q^i_n(t) + q^i_n(t)] = \mathcal{O}\left(\frac{\log^2(\epsilon)}{\sqrt{\epsilon}}\right), \tag{25}$$

where $\mathbf{x}^t(\boldsymbol{\gamma}^t)$ denotes the real-time operations obtained from the Lagrangian minimization (22a).

Theorem 1 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 [9], [10]. Comparing with the standard tradeoff $[\epsilon, 1/\epsilon]$ under Algorithm 1, the learn-and-adapt design of Algorithm 2 remarkably improves the delay performance.

V. Numerical Tests

Numerical tests are provided to confirm our analytical claims and demonstrate the merits of the proposed algorithms. Two types of network services are considered on the platform with $N = 7$ VMs. The first type of network service is $\{f_1, f_2, f_3\}$ and the second type of network service is $\{f_3, f_1, f_2\}$. $\mathcal{N}_1 = \{1, 2\}$, $\mathcal{N}_2 = \{3, 4\}$ and $\mathcal{N}_3 = \{5, 6, 7\}$. The processing and routing prices α^t_n and $\beta^t_{[a,b]}$ are uniformly distributed over $[0.1, 1]$ by default. $\delta^{\max}_n$ and $u^{\max}_{[a,b]}$ are generated from a uniform distribution within $[10, 20]$. R^i_n is uniformly distributed $[1, 5]$. 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 $\boldsymbol{\theta} = 2\sqrt{\epsilon}\log^2(\epsilon)$. Apart from the proposed Algorithms 1 and 2, we also simulate a heuristic algorithm (Heu) as the

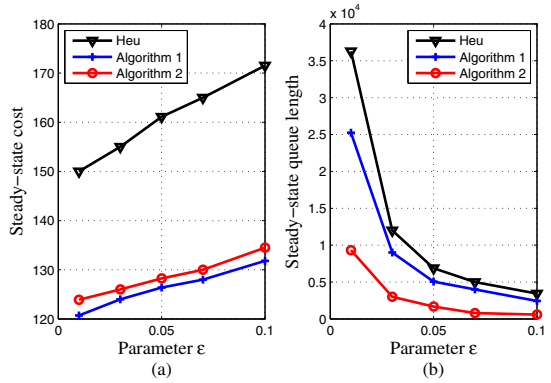

Fig. 2. Comparison of steady-state costs and queue lengths (after 10^4 slots).

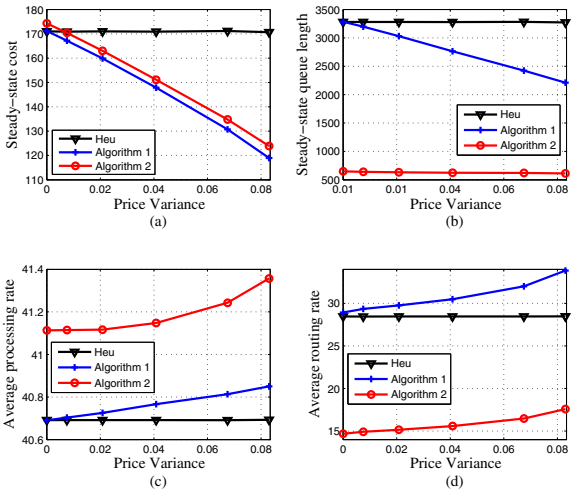

Fig. 3. Comparison of steady-state costs, queue lengths, processing and routing rates (after 10^4 slots).

benchmark, which decides the processing and routing rates only based on queue differences, with no price considerations.

Fig. 1 compares the three algorithms in terms of the time-average cost and the instantaneous queue length, as the time elapses. It can be seen from Fig. 1(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 30.0% larger. We can also see that Algorithm 2 exhibits faster convergence than Algorithm 1 and Heu, as its time-average cost quickly researches the optimal steady-state value by leveraging the learning process. Fig. 1(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 75.7% and 82.7% 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. 2 compares the steady-state cost and queue length of the three algorithms, with the growth of the 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 Propositions 1 and 2, and Theorem 1.

The steady-state cost and queue length are also compared under different price variances in Figs. 3(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 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. 3(c) and (d), 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.

VI. 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 of processing and routing VNFs 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 $[\epsilon, \log^2(\epsilon)/\sqrt{\epsilon}]$.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] T. Enokido and M. Takizawa, "An energy-efficient load balancing algorithm for virtual machine environments to perform communication type application processes," in *Proc. IEEE AINA*, 2016.

[6] Z. Á. Mann, "Multicore-aware virtual machine placement in cloud data centers," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3357–3369, Nov. 2016.

[7] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proc. IEEE CloudNet*, 2015.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.,* to appear, Aug. 2017.

[13] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, R. P. Liu, and G. B. Giannakis, "Distributed online optimization of network function virtualization under stochastic arrivals of network service chains," *IEEE J. Sel. Areas Commun.,* submitted, Mar. 2017.