Distributed Placement and Online Optimization of Virtual Machines for Network Service Chains

Xiaojing Chen[1,3], Wei Ni[2], Iain B. Collings[3], Xin Wang[1], and Shugong Xu[4]
[1]Key Lab of EMW Information (MoE), Dept. of Commun. Sci. & Engr., Fudan University, China
[2]Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia
[3]School of Engineering, Macquarie University, Australia
[4]Shanghai Institute for Advanced Communication and Data Science, Shanghai University, China.

***Abstract*—This paper proposes a new fully decentralized approach for online placement and optimization of virtual machines (VMs) for network functions virtualization (NFV). The approach is non-trivial as the virtual network functions (VNFs) constituting network services must be executed correctly in order at different VMs, coupling the optimal decisions of VMs on processing or forwarding. Leveraging Lyapunov optimization techniques, we decouple the optimal decisions by minimizing the instantaneous NFV cost in a distributed fashion, and achieve the asymptotically minimum time-average cost. We also reduce the queue length by allowing individual VMs to (un)install VNFs based on local knowledge, adapting to the network topology and the temporal and spatial variations of services. Simulations show that the proposed approach is able to reduce the time-average cost of NFV by 71% and reduce the queue length (or delay) by 74%, as compared to existing approaches.**

***Index Terms*—Virtual machine, network functions virtualization, Lyapunov optimization.**

I. Introduction

Decoupling dedicated hardware from network services and replacing it with programmable virtual machines (VMs), network functions virtualization (NFV) is able to provide critical network functions on top of optimally shared physical infrastructure. This can avoid disproportional hardware investments on short-lived functions, and adapt quickly as network functions evolve [1]. A network service can consist of multiple elemental Virtual Network Functions (VNFs) which need to operate in a predefined order at different VMs running the VNFs [2]. The VNFs deployed over VMs are connected through virtual links to adaptively create a service chain in response to a demand for the service (i.e., workload).

Challenges arise from optimal decision-makings of processing or routing service requests at each VM, especially in large-scale network platforms that require distributed decision-makings and operations. On one hand, given the order of VNFs to process the request of a network service, the optimal decisions that individual VMs make in a distributed fashion are coupled. On the other hand, stochasticity prevails in the arrivals of network services and the link capacity between VMs stemming from concurrent traffic [3]. Prices can also vary for the operations that a VM provides, depending on the pricing policy of the infrastructure providers. Other challenges also include the placement of VMs in adaptation to the changing demand from network services.

Existing works have been focused on the placement of VMs under the assumption of persistent arrivals of service requests (or workloads) [4], or full knowledge on the statistics of the arrivals [5]. Genetic heuristic approaches were developed to solve mixed integer linear programming (MILP) to minimize the delay of network service chains [6]. Greedy algorithms were developed to minimize flowtime or cost, or maximize revenue at a snapshot of the network [7]. These heuristic schemes still need to run in a centralized manner, limiting scalability.

This paper proposes a new fully distributed approach for automated placement and online optimization of VMs. Capturing random service arrivals, time-varying prices, and sequentially bonded service chains, a new stochastic method is developed to decouple optimal decision-makings across different VMs and different time slots. Leveraging Lyapunov optimization techniques, closed-form optimal solutions are derived per slot to asymptotically minimize the time-average cost of NFV while stabilizing the network. The contributions of the paper can be summarized as follows.

1) We reduce the number of workload queues per VM to the number of VNFs, as opposed to the number of network service types (as is typically required). As a result, the queue management can be simplified and control signaling can be substantially reduced.
2) Accommodating multiple VNFs per VM, we derive asymptotically optimal distributed decisions to select services to be processed and to be forwarded per slot, minimizing the time-average cost of NFV and stabilizing the network in infinite time horizon.
3) Based on local knowledge of queue lengths, a new distributed protocol is developed to adaptively install and/or uninstall VNFs at each VM to reduce processing backlog and cost.

Corroborated by simulations, the proposed approach is able to reduce the time-average cost of NFV by 71% and the queue length by 74%, as compared to existing approaches.

In a different yet relevant context, stochastic optimization,

†Work in this paper was supported by the National Natural Science Foundation of China grant 61671154 and the Innovation Program of Shanghai Municipal Education Commission.

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

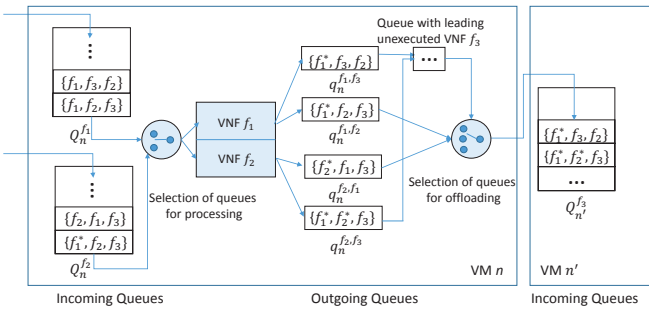

Fig. 1. An illustration on the processing and routing procedure of network services within VM n, where f^* indicates executed VNFs.

also known as Lyapunov optimization, has been developed for resource allocation, routing and service computing in queueing systems to combat stochastic arrivals of workloads or energy [8]–[12]. However, none of the existing approaches [8]–[12] have taken sequentially chained network services or online placement of VMs into account. Distinctively different from the existing approaches, our approach is able to optimize both the processing and offloading of chained network services, redeploy VNFs in real-time. Moreover, our approach can reduce the number of queues per VM to be dramatically less than the number of service types, without compromising the validity of the stochastic optimization framework and the asymptotic optimality of the solutions.

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 service processing and routing, and the automated placement of VMs are developed. Performance guarantee is given in Section IV. Numerical tests are provided in Section V, followed by concluding remarks in Section VI.

II. System Model

Consider a platform which consists of N VMs and runs K VNFs, namely, $f_1, \ldots, f_K$; let $\mathcal{F} = \{f_1, \ldots, f_K\}$. Let $\mathcal{I} = \{1, \ldots, I\}$ collect all possible network service chains, each of which can be denoted as a differently permuted sequence of $\{f_1, \ldots, f_K\}$. By installing corresponding software, each VM can run multiple VNFs. Let $\mathcal{F}_n$ collect the VNFs that VM n runs. Assume that every VM can admit network service chains, and output the results; see Fig. 1. In this sense, a request of network service chain needs to traverse among VMs, until all the VNFs of the service chain run correctly in order. Let $\mathcal{N} = \{1, \ldots, N\}$ collect the N VMs and $\mathcal{L} = \{[a,b], a, b \in \mathcal{N}\}$ collect L directional links between the VMs. Let U_n denote the set of the upstream VMs of VM n, and D_n the set of the downstream VMs. Let α_{ab}^t denote the price for forwarding network service chains over link $[a,b]$ at time t, and $\beta_n^{f_k,t}$ the price for VM n to process network service chains with leading unexecuted VNF f_k at time t.

We design up to $Y(n) + Y(n)K$ queues per VM n, where $Y(n)$ is the number of VNFs that VM n can run. Each of the $Y(n)$ queues buffers incoming network services with a leading unexecuted VNF $f_k \in \mathcal{F}$. Each of the other $Y(n)K$ queues buffers the outgoing results of the $Y(n)$ incoming queues after VNF f_k is run, and is to be routed to other VMs.

Per slot t, let $Q_n^{f_k}(t)$ denote the queue length of network service chains with a leading unexecuted VNF f_k at VM n, and $q_n^{f_k,f_{k'}}(t)$ denote the queue length of network service chains with VNF f_k just run at VM n and a new leading unexecuted VNF $f_{k'}$. Let $\mathbf{Q}(t) = \{Q_n^{f_k}(t), \forall n \in \mathcal{N}, f_k \in \mathcal{F}\}$ and $\mathbf{q}(t) = \{q_n^{f_k,f_{k'}}(t), \forall n \in \mathcal{N}, f_k, f_{k'} \in \mathcal{F}\}$. Let $R_n^{f_k,t} \leq R^{\max}$ denote the arrival rate (in services per slot) of network services with a leading unexecuted VNF f_k at VM n, where $R^{\max}$ is the maximum arrival rate.

For generality, we consider that any VM n or directional link $[a,b]$ can process or transmit more than one network service during a slot. The aggregate data rate (in services per slot) of network service chains with a leading unexecuted VNF $f_{k'}$ over link $[a,b]$ is denoted by $u_{ab}^{f_{k'}}(t) := \sum_{f_k} u_{ab}^{f_k,f_{k'}}(t)$, where $u_{ab}^{f_k,f_{k'}}(t)$ is the data rate of network service chains with VNF f_k just executed at VM a and a new leading unexecuted VNF $f_{k'}$. It is easy to see:

$$\sum_{f_{k'}} \sum_{f_k} u_{ab}^{f_k,f_{k'}}(t) = \sum_{f_{k'}} u_{ab}^{f_{k'}}(t) \leq u_{ab}^{\max},$$
$$u_{ab}^{f_k,f_{k'}}(t) \geq 0, \ \forall [a,b], f_k, f_{k'} \tag{1}$$

where $u_{ab}^{\max}$ is the maximum data rate over link $[a,b]$.

Let $\delta_n^{f_k}(t) := \sum_{f_{k'}} \delta_n^{f_k,f_{k'}}(t)$ denote the aggregate processing rate (in services per slot) of network service chains with a leading unexecuted VNF f_k at VM n, where $\delta_n^{f_k,f_{k'}}(t)$ is the processing rate of network service chains with a leading unexecuted VNF f_k, followed by $f_{k'}$. We have

$$\sum_{f_k} \sum_{f_{k'}} \delta_n^{f_k,f_{k'}}(t) = \sum_{f_k} \delta_n^{f_k}(t) \leq \delta_n^{\max},$$
$$\delta_n^{f_k,f_{k'}}(t) \geq 0, \ \forall n, f_k, f_{k'} \tag{2}$$

where $\delta_n^{\max}$ is the maximum processing rate of VM n.

Therefore, the queue length of incoming network service chains with a leading unexecuted VNF f_k at VM n follows, $\forall n, f_k, t$,

$$Q_n^{f_k}(t+1) = Q_n^{f_k}(t) + \sum_{a \in U_n} u_{an}^{f_k}(t) + R_n^{f_k,t} - \delta_n^{f_k}(t), \tag{3}$$

with $\delta_n^{f_k}(t) \leq Q_n^{f_k}(t)$ due to data causality.

The queue length of outgoing network service chains with VNF f_k run at VM n, followed by an unexecuted VNF $f_{k'}$, satisfies, $\forall n, f_k, f_{k'}, t$,

$$q_n^{f_k,f_{k'}}(t+1) = q_n^{f_k,f_{k'}}(t) + \delta_n^{f_k,f_{k'}}(t) - \sum_{d \in D_n} u_{nd}^{f_k,f_{k'}}(t), \tag{4}$$

where $\sum_{d \in D_n} u_{nd}^{f_k,f_{k'}}(t) \leq q_n^{f_k,f_{k'}}(t)$ is also due to causality. Note that $q_n^{f_k,f_{k'}}(t) = 0$ in the case that f_k is the last VNF of a network service chain (i.e., $f_{k'} \in \emptyset$); namely, all results are executed and output at VM n.

We can define the network is *stable* if and only if [13]:

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

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

We also define the total cost of routing services and processing VNFs at all VMs as per slot t, as given by:

$$\Phi(u_{ab}^{f_k}(t),\delta_n^{f_k}(t)):=\sum_{a,b,f_k}f(u_{ab}^{f_k}(t))+\sum_{n,f_k}h(\delta_n^{f_k}(t))), \tag{6}$$

where $f(u_{ab}^{f_k}(t))=\alpha_{ab}^t u_{ab}^{f_k}(t)$ and $h(\delta_n^{f_k}(t))=\beta_n^{f_k,t}\delta_n^{f_k}(t)$ are the costs that the network service provider charges over the usages of links and VMs, respectively.

III. Distributed Online Processing, Routing and Placement

Our objective is to minimize the time-average cost of NFV while preserving network stability (i.e., finite queue lengths), under random network service arrivals and time-varying prices. This is to be achieved by making stochastically optimal decisions on processing or routing network services at every VM and automating the placement of VMs in a distributed fashion. Given multiple VNFs installed per VM, the decisions that every VM needs to make include choosing VNFs to run unexecuted service chains, and choosing executed service chains to route for further processing.

A. Problem Formulation and Transformation

We formulate the problem of interest from the perspective of long-term time-average cost. Let $\mathbf{x}^t:=\{u_{ab}^{f_k}(t),\forall[a,b],f_k;\delta_n^{f_k}(t),\forall n,f_k\}$ and $\mathcal{X}:=\{\mathbf{x}^t,\forall t\}$. The problem of interest can be written as

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

where the expectation, $\mathbb{E}[\Phi(\mathbf{x}^t)]$, is taken over all randomnesses. Concatenate the random parameters into a state vector $\boldsymbol{s}^t:=\{R_n^{f_k,t},\alpha_{ab}^t,\beta_n^{f_k,t},\forall[a,b],n,f_k\}$. Suppose that $\boldsymbol{s}^t$ is independent and identically distributed (i.i.d.) across slots.

Minimizing the time-average cost over an infinite time horizon in (7) is challenging. In particular, the queue dynamics in (3) and (4) couple the optimization variables in time, rendering intractability for traditional solvers. In this paper, we decouple the variables over time by capitalizing the Lyapunov optimization method, which is proven to asymptotically minimize the time-average cost while preserving network stability. It is worth mentioning that none of existing Lyapunov optimization algorithms (e.g., in [8]–[12]) can be used or extended to solve the problem of interest, since the chained network services lead to a new problem and require new transformations and solutions.

A quadratic Lyapunov function $L(t)$ can be defined to measure the stability of the queues, as given by

$$L(t):=\frac{1}{2}[\sum_{n,f_k}Q_n^{f_k}(t)^2+\sum_{n,f_k,f_{k'}}q_n^{f_k,f_{k'}}(t)^2]. \tag{8}$$

The function becomes large, as the queueing system moves towards unstable states. The stability of the network can be achieved by taking control actions that harness the Lyapunov function at any slot.

A drift-plus-penalty can be specified by $\triangle_V(t):=\triangle(t)+V\mathbb{E}[\Phi(\mathbf{x}^t)]$, where $\triangle(t):=\mathbb{E}[L(t+1)-L(t)]$ is the Lyapunov drift measuring the difference of the Lyapunov function between two consecutive slots. Here, $\Phi(\mathbf{x}^t)$ is the time-average cost and V is a predefined non-negative weight to tune the tradeoff between the queue lengths and cost. By minimizing the upper bound of the drift-plus-penalty at every slot, we decouple the variables in time, and asymptotically minimize the time-average cost $\Phi(\mathbf{x}^t)$ while stabilizing the queues (or in other words, the delays of delivering services). The upper bound of $\triangle_V(t)$ can be obtained in the following lemma[1].

Lemma 1. *For any queues and actions, $\triangle_V(t)$ is upper bounded by*

$$\begin{aligned}\triangle_V(t)\leq B+\mathbb{E}\Big(V\Phi(\mathbf{x}^t)+\sum_{n,f_k}Q_n^{f_k}(t)[\sum_{a\in U_n}u_{an}^{f_k}(t)+R_n^{f_k,t}\\ -\delta_n^{f_k}(t)]+\sum_{n,f_k,f_{k'}}q_n^{f_k,f_{k'}}(t)[\delta_n^{f_k,f_{k'}}(t)-\sum_{d\in D_n}u_{nd}^{f_k,f_{k'}}(t)]\Big),\end{aligned} \tag{9}$$

where $B:=\frac{2NK+NK^2}{2}N^{\max}u_{ab}^{\max 2}+NKR^{\max 2}+\frac{NK+NK^2}{2}\delta_n^{\max 2}$.

Problem (7) is now relaxed to minimizing the right-hand side of (9) at each time slot t, subject to the instantaneous constraints (1) and (2). Let $\mathcal{X}^t$ denote the set of $\{u_{ab}^{f_k}(t),\forall[a,b],f_k;\delta_n^{f_k}(t),\forall n,f_k\}$ satisfying constraints (1) and (2) per t. The relaxed problem is then given as follows:

$$\begin{aligned}\min_{\mathbf{x}^t\in\mathcal{X}^t}V\Phi(\mathbf{x}^t)+\sum_{n,f_k}\{Q_n^{f_k}(t)[\sum_{a\in U_n}u_{an}^{f_k}(t)+R_n^{f_k,t}-\delta_n^{f_k}(t)]\\ +\sum_{f_{k'}}q_n^{f_k,f_{k'}}(t)\delta_n^{f_k}(t)-\sum_{f_{k''}}q_n^{f_{k''},f_k}(t)\sum_{d\in D_n}u_{nd}^{f_k}(t)\}.\end{aligned} \tag{10}$$

Through rearrangement, (10) is equivalent to

$$\begin{aligned}\min_{\mathbf{x}^t\in\mathcal{X}^t}\sum_{n,f_k,d\in D_n}[V\alpha_{nd}^t-(\sum_{f_{k''}}q_n^{f_{k''},f_k}(t)-Q_d^{f_k}(t))]u_{nd}^{f_k}(t)\\ +\sum_{n,f_k}[V\beta_n^{f_k,t}-(Q_n^{f_k}(t)-\sum_{f_{k'}}q_n^{f_k,f_{k'}}(t))]\delta_n^{f_k}(t).\end{aligned} \tag{11}$$

Solving (11) in general requires centralized control. Nevertheless, the VMs and links here do not interfere with each other; i.e., (11) can be decoupled among VMs. In other words, each VM can minimize (11) locally, given the queue length

[1]The proofs for all lemmas and theorems are omitted due to limited space, and can be found in the extended journal version [14].

information of its neighboring VMs. Solving (11) is equivalent to solving the following problem per VM n:

$$\min_{\mathbf{x}^t \in \mathcal{X}^t} \sum_{f_k, d \in D_n} W_{nd}^{f_k} u_{nd}^{f_k}(t) + \sum_{f_k} W_n^{f_k} \delta_n^{f_k}(t), \tag{12}$$

where we define the following price-queue weights:

$$W_{nd}^{f_k} := V\alpha_{nd}^t - (\sum_{f_{k''}} q_n^{f_{k''}, f_k}(t) - Q_d^{f_k}(t));$$
$$W_n^{f_k} := V\beta_n^{f_k,t} - (Q_n^{f_k}(t) - \sum_{f_{k'}} q_n^{f_k, f_{k'}}(t));$$
$$\forall n, f_k, d \in D_n. \tag{13}$$

Problem (12) is linear programming in the form of weighted-sum minimization, and can be readily solved by using existing standard solvers, e.g., the interior-point method. The complexity is typically polynomial around $\mathcal{O}(N^3)$.

B. Online Operation of VMs

We propose an optimal distributed online processing and routing solution for (12), as shown in Algorithm 1. Particularly, per slot, a VM can prioritize the queues with different leading VNFs, and process or route the queue with the highest priority. The queues with smaller price-queue weights are given higher priorities. Each price-queue weight contains price information and difference of queue pairs. For example, $(\sum_{f_{k''}} q_n^{f_{k''}, f_k}(t) - Q_d^{f_k}(t))$ denotes the difference of the sum of all outgoing queues with leading unexecuted VNF f_k at VM n, and the incoming queue with the same leading VNF at the downstream VM d. With a large queue difference, the outgoing queues buffered at VM n can be routed to the corresponding incoming queue at VM d, pushing the difference to zero. Likewise, the queue difference $(Q_n^{f_k}(t) - \sum_{f_{k'}} q_n^{f_k, f_{k'}}(t))$ also decides whether the incoming queue with a leading unexecuted VNF f_k is chosen to be processed at VM n.

Note that Algorithm 1 is fully distributed, since every VM only needs to know the queue lengths of its own and its immediate neighbors. The decisions of a VM, locally made by comparing the queue-price weights, comply with (3) and (4), and therefore preserve the asymptotic optimality of the entire network, as to be shown in Theorems 1 and 2.

C. Proposed Automated Placement of VMs

We proceed to propose to reduce the queue lengths (delays) by adaptively placing and updating VNFs across the network. This is because heavily loaded VMs build up long queues and degrade the whole performance of the network. Specifically, every VM collates the lengths of its own incoming queues and its immediate one-hop neighbors' queues. If a VM has the shortest incoming queue of all its neighbors, it can uninstall the VNF that is the leading unexecuted VNF buffered in the shortest incoming queue. Likewise, every VM also collates the lengths of its own outgoing queues and its immediate one-hop neighbors' queues. If a VM has the longest outgoing queue of all its neighbors, it can install the VNF that is the leading unexecuted VNF buffered in the longest outgoing queue. This can help reduce the average length of queues in the network and therefore the delay of processing.

Algorithm 1 Distributed Online Optimization of NFV

1: At time slot t, each VM n observes its incoming and outgoing queue lengths, $Q_n^{f_k}(t)$ and $q_n^{f_k, f_{k'}}(t)$, and the incoming queues of its downstream VMs, $Q_d^{f_{k'}}(t), \forall d \in D_n$.
2: Find $\{f_k^*\} = \arg\min_{f_k} W_n^{f_k}$. If $W_{n,}^{f_k^*} < 0$, set $\delta_n^{f_k^*}(t) = \min\{\delta_n^{\max}, Q_n^{f_k^*}(t)\}$. That is, use the full capacity of VM n to process the incoming queue of network services with a leading unexecuted VNF f_k^*, if the queue is sufficiently long (i.e., longer than $\delta_n^{\max}$) and maintains the minimum price-queue weight $W_n^{f_k}$. Otherwise if $W_n^{f_k^*} \geq 0$, set $\delta_n^{f_k^*}(t) = 0$. The processing rates $\{\delta_n^{f_k^*, f_{k'}}(t), \forall f_{k'}\}$ of different network service chains are readily derived with all queues following FIFO (first in, first out).
3: Find $\{f_k^*, d^*\} = \arg\min_{f_k, d} W_{nd}^{f_k}$. If $W_{nd^*}^{f_k^*} < 0$, set $u_{nd^*}^{f_k^*}(t) = \min\{u_{nd^*}^{\max}, q_n^{f_k^*}(t)\}$. That is, allocate the full rate over link $[n, d^*]$ to the outgoing queue of network services with a leading unexecuted VNF f_k^*, if the queue is sufficiently long (i.e., longer than $u_{nd^*}^{\max}$) and achieves the minimum price-queue weight $W_{nd}^{f_k}$ over the link. Otherwise if $W_{nd^*}^{f_k^*} \geq 0$, set $u_{nd^*}^{f_k^*}(t) = 0$. Similarly, the routing rates $\{u_{nd^*}^{f_{k''}, f_k^*}(t), \forall f_{k''}\}$ of different network service chains are readily derived with all queues following FIFO.
4: Repeat Steps 2-3 until either the processing capability of the VM and all its outgoing links are fully utilized, or there are no queues to be scheduled.
5: Update $Q_n^{f_k}(t)$ and $q_n^{f_k, f_{k'}}(t)$ for all queues according to (3) and (4).

Such automated placement and update of VMs can be implemented in a fully distributed fashion. Particularly, all the immediate neighbors of a VM, which installs a new VNF to reduce its longest outgoing queue or uninstall an existing VNF to reduce its process cost, would not install or uninstall VNFs, given the consistent knowledge collated at each of the nodes. A distributed protocol can be summarized in Algorithm 2, which can be run at a much large interval of tens to hundreds time slots.

IV. Performance Guarantee

Relying on Lyapunov optimization techniques, the asymptotic optimality of a stabilized network can be formally established as follows:

Theorem 1. *If $\boldsymbol{s}^t$ is i.i.d. over slots, then the proposed Algorithm 1 yields a feasible dynamic control scheme for* (7)*, which is asymptotically near-optimal in the sense that*

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

Algorithm 2 Automated Placement of VMs

1: Run Algorithm 1 with initially installed VNFs until it converges.
2: At every VM, find the shortest incoming queue of itself and all its neighbors, and uninstall the VNF which corresponds to the shortest incoming queue, (i.e., the shortest queue with the VNF as the leading unexecuted VNF), if the queue belongs to the VM.
3: At every VM, find the longest outgoing queue of itself and all its neighbors, and install the VNF which corresponds to the longest outgoing queue, (i.e., the longest queue with the VNF as the leading unexecuted VNF), if the queue belongs to the VM.
4: Repeat Steps 1-3 until the total steady-state queue lengths (delays) converge.

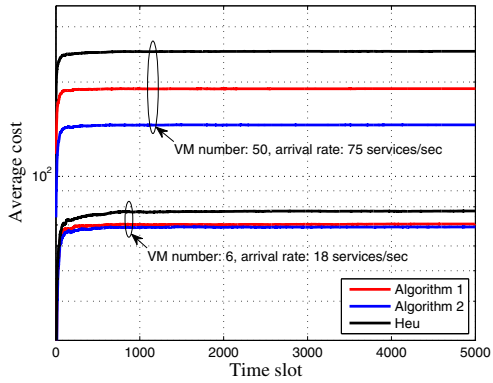

Fig. 2. Comparison of the evolution of time-average costs.

where $\Phi^*(\mathbf{x}^t)$ *denotes the resultant cost with the proposed algorithm, and* Φ^{opt} *is the optimal value of* (7) *under any possible control policy (i.e., the processing and routing decisions per VM), even if that relies on knowing future realizations of random variables.*

Theorem 2. *Assume that there exists a stationary policy* $\mathcal{P}^{stat}$ *and* $\mathbb{E}[\sum_{a\in U_n} u_{an}^{f_k}(t) + R_n^{f_k,t} - \delta_n^{f_k}(t)] \leq -\zeta$, *and* $\mathbb{E}[\delta_n^{f_k,f_{k'}}(t) - \sum_{d\in D_n} u_{nd}^{f_k,f_{k'}}(t)] \leq -\zeta$, *where* $\zeta > 0$ *is a slack vector constant. Then, all queues are stable and the time-average queue length satisfies:*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{n,f_k,f_{k'}} \mathbb{E}[Q_n^{f_k}(t) + q_n^{f_k,f_{k'}}(t)] = \mathcal{O}(V). \quad (14)$$

Theorems 1 and 2 assert that the stochastically minimized time-average cost obtained by the proposed distributed method converges to a region within an optimality gap of $\mathcal{O}(\frac{1}{V})$, which vanishes as $V \to \infty$. The typical tradeoff from the stochastic network optimization holds in this case [13]: an $\mathcal{O}(V)$ queue length is necessary, when an $\mathcal{O}(\frac{1}{V})$ close-to-optimal cost is achieved.

V. Numerical Tests

Numerical tests are provided to confirm our analytical claims and demonstrate the merits of the proposed algorithms.

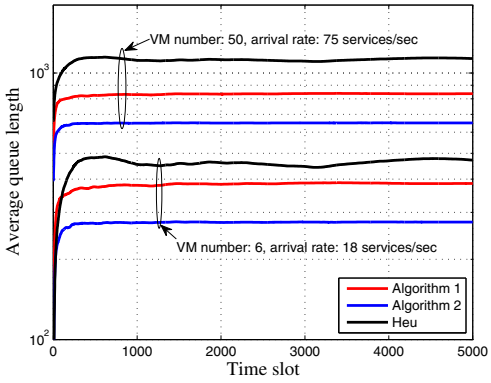

Fig. 3. Comparison of the evolution of time-average queue lengths.

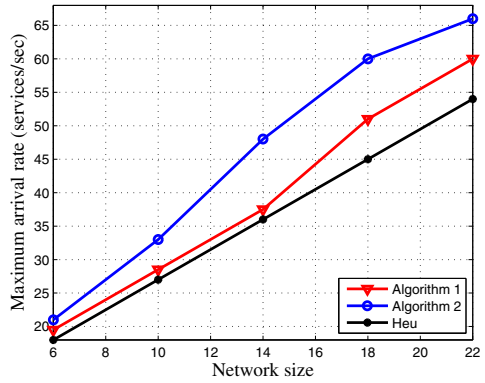

Fig. 4. Comparison of maximum arrival rates, where $V = 20$.

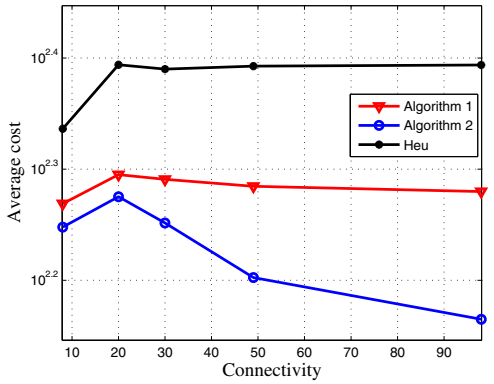

Fig. 5. Comparison of steady-state costs under different network connectivities, where $N = 50$, $V = 20$, and average arrival rate is 75 services/sec.

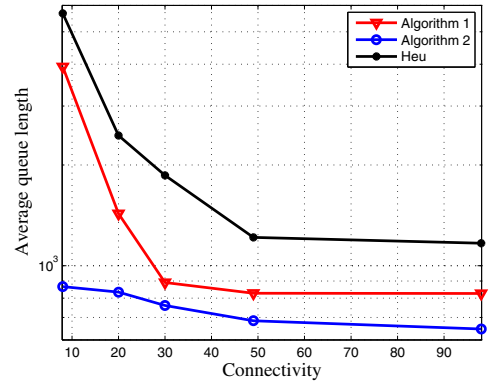

Fig. 6. Comparison of steady-state queue lengths under different network connectivities, where $N = 50$, $V = 20$, and average arrival rate is 75 services/sec.

Two types of network services are considered on the platform with $N = 6$ VMs; unless otherwise specified. The connectivity of the platform is model as a complete graph. The default arrival rate of network service requests is uniformly distributed with a mean of 18 services/sec. The first type of network service is $\{f_1, f_2, f_3, f_4\}$ and the second type of network service is $\{f_2, f_4, f_3, f_1\}$. Each of the VMs is allowed to install up to two VNFs, and is initialized by $\mathcal{F}_1 = \{f_1, f_2\}$, $\mathcal{F}_2 = \{f_1, f_3\}$, $\mathcal{F}_3 = \{f_3, f_4\}$, $\mathcal{F}_4 = \{f_2, f_3\}$, $\mathcal{F}_5 = \{f_1, f_4\}$ and $\mathcal{F}_6 = \{f_2, f_3\}$. The routing and processing prices α_{ab}^t and $\beta_n^{f_k,t}$ are uniformly distributed over $[0.1, 0.5]$ by default; unless otherwise specified. Set $u_{ab}^{\max} = 20$ and $\delta_n^{\max} = 20$. The tradeoff variable is $V = 20$ by default. In addition to the proposed Algorithms 1 and 2, we also simulate a heuristic algorithm (Heu) as the benchmark, which chooses the queue with the largest incoming queue length to process at each VM per slot, and makes routing decisions based merely on queue differences, rather than price-queue weights, as done in Algorithm 1. In other words, the decisions are made without considering processing and routing prices, or V.

Figs. 2 and 3 compare the evolution of the time-average costs and queue lengths of Algorithms 1 and 2, and Heu. It can be seen from Fig. 2 that the time-average cost of Heu converges to the highest value among the three algorithms. Specifically, the average cost of Heu is about 32% and 71% higher than those of Algorithm 1 and Algorithm 2, respectively, when the network size (i.e., the number of VMs) $N = 50$ and the arrival rate is 75 services/sec. Fig. 3 shows that Heu also incurs the largest queue lengths among the three algorithms. Particularly, the aggregated queue length of Heu is about 36% and 74% larger than those of Algorithm 1 and Algorithm 2 when the network size $N = 50$, respectively.

We further see in Figs. 2 and 3 that the proposed automated placement of VMs described in Algorithm 2 does not increase the cost of the network, which is able to reduce the queue lengths efficiently and significantly, as compared to the proposed online optimization of VMs described in Algorithm 1. In other words, the placement of VMs is particularly interesting to bring down queue length or delay in NFV.

Fig. 4 illustrates the maximum arrival rates (i.e., throughputs) of Algorithms 1 and 2, and Heu, with the growth of network size. It can be observed that the maximum arrival rates of service requests increase as the network size grows. Specifically, Algorithm 2 tolerates the largest arrival rate of services while stabilizing the network among the three algorithms. The benefit of automated placement of VMs on delay reduction is further demonstrated.

The comparison of steady-state costs and queue lengths under different network connectivities is further plotted in Figs. 5 and 6. We can see the costs first increase and then decrease, and the queue lengths keep decreasing, as the network connectivity increases. This is because when the connectivity is very small, little routing is conducted, which reduces costs but enlarges queue lengths. As the connectivity is large enough, Algorithms 1 and 2 are able to take the advantage of price imbalance between different links and make the most cost-effective decisions. Particularly, the automated placement of VMs, captured in Algorithm 2, works efficiently in reducing queue length when the connectivity is small, revealing its superiority in reducing the delay of NFV.

VI. Conclusion

In this paper, novel distributed online processing of network service chains was developed to minimize the time-average cost of NFV while stabilizing the network. By exploiting Lyapunov optimization techniques, asymptotically optimal decisions of processing and routing network services were instantly generated at every VM, adapting to the network topology and network service arrivals. Automated placement of VNFs was proposed to reduce the queue lengths of unexecuted network services, achieving stable adaptive deployment of VNFs against the connectivity of individual VMs and the spatial distributions of services.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, Apr. 2015.

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

[9] 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.

[10] 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*, to appear, 2017.

[11] 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.

[12] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, to appear, 2017.

[13] 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.

[14] X. Chen, W. Ni, I. B. Collings, X. Wang, and S. Xu, "Automated function placement and online optimization of network functions virtualization," *IEEE Trans. Commun.*, submitted, Jan. 2018.