Online Scaling of NFV Service Chains Across Geo-Distributed Datacenters

Yongzheng Jia, Chuan Wu, *Senior Member, IEEE, Member, ACM*, Zongpeng Li, Franck Le, and Alex Liu

***Abstract*— Network Function Virtualization (NFV) is an emerging paradigm that turns hardware-dependent implementation of network functions (i.e., middleboxes) into software modules running on virtualized platforms, for significant cost reduction and ease of management. Such virtual network functions (VNFs) commonly constitute service chains, to provide network services that traffic flows need to go through. Efficient deployment of VNFs for network service provisioning is a key to realize the NFV goals. Existing efforts on VNF placement mostly deal with offline or one-time placement, ignoring the fundamental, dynamic deployment and scaling need of VNFs to handle practical time-varying traffic volumes. This work investigates dynamic placement of VNF service chains across geo-distributed datacenters to serve flows between dispersed source and destination pairs, for operational cost minimization of the service chain provider over the entire system span. An efficient online algorithm is proposed, which consists of two main components: 1) A regularization-based approach from online learning literature to convert the offline optimal deployment problem into a sequence of one-shot regularized problems, each to be efficiently solved in one time slot and 2) An online dependent rounding scheme to derive feasible integer solutions from the optimal fractional solutions of the one-shot problems, and to guarantee a good competitive ratio of the online algorithm over the entire time span. We verify our online algorithm with solid theoretical analysis and trace-driven simulations under realistic settings.**

***Index Terms*— Network function virtualization, service chains, online algorithm, convex optimization.**

I. Introduction

TRADITIONALLY, network functions such as firewalls, proxies, network address translators (NATs) and intrusion detection systems (IDSs) are implemented by dedicated hardware middleboxes, which are costly and difficult to scale. The emerging paradigm of network function virtualization (NFV) aims to revolutionize network function provisioning by running the respective software in standard virtualized platforms, *e.g.*, virtual machines (VMs) on industry-standard servers, in order to achieve fundamental flexibility in deployment and management, as well as significant cost reduction [1].

To realize these goals, a fundamental challenge is to strategically deploy the virtualized network functions (VNFs), in terms of the number of instances for each VNF and the deployment locations, and to dynamically scale the deployment, by adding/removing instances of VNFs with increase/decrease of the demand. In particular, the network functions are typically connected to compose service chains [2] that provide different network services [3], [4]. Along a service chain, network traffic flows are required to go through multiple stages of network function processing in a particular order. A service chain can be placed within one datacenter (*e.g.*, the service chain "Firewall→IDS→Proxy" providing a company's access service is typically deployed in an on-premise datacenter), or distributed across multiple datacenters, (*e.g.*, for the virtualization of WAN optimizers, IP Multimedia Subsystem (IMS), mobile core networks [1]). For the example of WAN optimization, deduplication or compression functions are deployed close to the sender of a flow and traffic shaping can happen anywhere along the route from the sender to the receiver. For virtualization of a control plane service chain in an IMS [5], namely "P-CSCF→S-CSCF", instances of P-CSCF, first contact of a user for call registration, should be distributed close to geo-dispersed callers, while S-CSCF for session control can be located more in the middle between a caller and a callee.

This study focuses on geo-distributed deployment of service chains and tackles the challenges of dynamic deployment and scaling of VNF instances in the chains. We take the perspective of a network service provider, who rents VMs from cloud datacenters owned by the respective datacenter or cloud operators, deploys VNFs on the VMs and assembles service chains for the usage of its flows. For example, the network service provider can provide service chains for WAN optimization consisting of different WAN optimizers, or can be an IMS provider who seeks to virtualize its control plane and data plane service chains (an example data plane service chain in an IMS is "Firewall→IDS→Transcoder").

We target cost minimization for the network service provider by optimally deploying geo-distributed service chains. The intriguing challenges are as follows. *First*, the deployment of VMs to run different network functions is not only relevant to VM rental cost, but also decides the bandwidth (cost) needed in-between datacenters to accommodate traffic flows passing through geo-dispersed network functions, which further decides quality of the network service, *e.g.*, end-to-end delay experienced by the flows. *Second*, when multiple

Manuscript received May 25, 2016; revised August 6, 2017 and December 15, 2017; accepted January 7, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor H. Jiang. Date of publication February 16, 2018; date of current version April 16, 2018. This work was supported in part by the Hong Kong RGC under Contract HKU 17204715, Contract 17225516, and Contract C7036-15G (CRF), in part by the NSFC under Grant 61628209, and in part by the HKU URC Matching Fund, Huawei, under Grant HIRP HO2016050002BE. *(Corresponding author: Chuan Wu.)*

Y. Jia and C. Wu are with The University of Hong Kong, Hong Kong (e-mail: jiayz13@mails.tsinghua.edu.cn; cwu@cs.hku.hk).

Z. Li is with the University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: zongpeng@ucalgary.ca).

F. Le is with the IBM Thomas J. Watson Research Center, Ossining, NY 10598 USA (e-mail: fle@us.ibm.com).

A. Liu is with Michigan State University, East Lansing, MI 48824, USA (e-mail: alexliu@cse.msu.edu).

Digital Object Identifier 10.1109/TNET.2018.2800400

service chains are in place, they may well share one or multiple common network functions, and may hence exploit the same collection of VNF instances deployed; this further demands deliberate computation of the geographical deployment of the shared instances, in order to balance the end-to-end performance experienced by different flows between various sources and destinations. *Last* but most importantly, we seek to make efficient online deployment and scaling decisions on the go with the variation of flows, while guaranteeing good performance over the long run of the system.

Existing efforts on VNF placement mostly deal with the offline or one-time placement, ignoring the fundamental, dynamic deployment and scaling demand of VNF service chains to handle practical time-varying traffic flows (see Sec. II for detailed discussions). In contrast, this work investigates dynamic placement of VNF service chains in geo-distributed cloud datacenters to serve dynamically-generated flows between various source/destination pairs across the globe, for service cost and delay minimization. We show that even in the offline setting, the problem we consider renders an NP-hard combinatorial nature, leading to significant difficulty in efficient online algorithm design. Looking deep into the structure of the problem, we design an efficient online algorithm based on the state-of-the-art online learning techniques and a well-designed dependent rounding scheme. Our detailed contributions are summarized as follows.

First, we formulate a practical online cost minimization problem enabling dynamic deployment and removal of VNF instances in different datacenters, as well as dynamic traffic flow routing among VNF instances in the respective service chains. Various deployment and running costs of VNF instances in different datacenters are considered, in addition to time-varying bandwidth costs to transmit flows into and out of a datacenter. As a key QoS performance indicator for network service provisioning, the average end-to-end delays of the flows are also formulated and minimized as part of our objective.

Second, we leverage a regularization based technique from the online learning literature [6] to transform the relaxation of the integer offline optimization problem into a sequence of regularized sub-problems. In particular, the regularization eliminates temporal correlation among decisions across time slots by lifting the precedence constraints coupling successive time slots into the objective function, such that each of the sub-problems can be efficiently and optimally solved in each time slot, using only information at the current time. By solving each sub-problem, VNF instance deployment/removal and flow routing decisions at the time are obtained in polynomial time, constituting part of the feasible (fractional) solution to the relaxed offline problem. Based on the KKT optimality conditions, we are able to show that an upper-bounded overall cost, as compared to the optimal offline solution, can be guaranteed by this (fractional) feasible solution. Moreover, we adapt the regularization framework into a general network flow model to handle the end-to-end delay.

Third, we carefully design an online randomized dependent rounding scheme for rounding fractional solutions (on the numbers of VNF instances in each datacenter) to feasible integer solutions of the original problem [7], [8]. Our online dependent rounding scheme consists of three modules: 1) A local clustering algorithm that groups datacenters into clusters with small intra-cluster end-to-end delays. One datacenter in each cluster with low resource costs is chosen as the "buffer" datacenter, to deploy VNF instances for absorbing un-served flows due to round-down in the number of VNF deployment in other datacenters. 2) An online dependent rounding algorithm which rounds the fractional numbers of VNF instances to integers while guaranteeing flow routing feasibility. 3) An optimal strategy for intra- and inter-cluster flow redirection based on the rounded solution. Our dependent rounding scheme balances well the minimization among different costs in our objective. Without relying on any future information, it together with the regularization based online algorithm guarantees a good competitive ratio in overall cost as compared to the offline optimum, which is insensitive to the total number of flows nor the number of time slots.

The rest of the paper is organized as follows. We discuss related work in Sec. II and present the problem model and the offline optimization problem in Sec. III. The online algorithm is given in Sec. IV and the dependent rounding scheme is discussed in Sec. V. We present trace-driven evaluation results in Sec. VI and conclude the paper in Sec. VII.

II. Related Work

Interest on NFV rippled out from a 2012 white paper [1] by telecommunication operators that introduced virtualized network functions running on commodity hardware. Recent IETF drafts presented the use cases of service chains applied across datacenters [2], [9], and highlighted the relation between service chains and NFV. Early efforts on NFV focused on bridging the gap between specialized hardware and network functions [10]–[12], and provided industrial standards for implementing network functions on VMs. Research activities from both industry and academia soon followed suit.

Several NFV management systems have been designed. SIMPLE [13] implements an SDN-based policy enforcement layer for efficient middlebox-specific "traffic steering" in datacenters. Clayman *et al.* [14] design an orchestrator-based architecture for automatic placement of the VNFs. Split/Merge [13] provides system support for achieving efficient, load-balanced elasticity when scaling in and out of virtual middleboxes. OpenNF [3] is a control plane that enables loss-free and order-preserving flow state migration across multiple instances of a VNF, in scenarios where flow packets are distributed across a collection of VNF instances for processing. Stratos [15] is an orchestration layer for efficient and scalable VNF provisioning and scaling via software-defined networking mechanisms. E2 [16] designs an application-agnostic scheduling framework to simplify deployment and scaling of VNFs for packet processing. These systems significantly facilitate the deployment of VNFs, and provide the system bases to support our deployment/scaling algorithm.

A fundamental issue in dynamic VNF provisioning is to optimally place, add and reduce the VNF instances with varying traffic, to provision needed service chains at

the minimal cost. There have been a few recent efforts on optimization algorithm design, which mostly deal with one-off placement, ignoring the dynamic nature of an NFV system. VNF-P [17] presents a one-time optimization model for VNF placement, considering hybrid deployment where part of the network service is provided by dedicated hardware and part by VNF instances, and designs a heuristic algorithm. Bari *et al.* [18] study a similar problem, formulate the problem into an integer linear program (ILP), and propose a dynamic programming based heuristic. Mehraghdam *et al.* [19] model a mixed integer quadratically constrained program (MIQCP) to pursue different optimization goals in VNF placement, without giving solution algorithms. Cohen *et al.* [20] investigate one-off VNF placement across different datacenters, to minimize the distance cost between clients and the VNFs that they need and the setup costs of these VNFs, and provide approximation algorithms with rigorous performance analysis. Zhang *et al.* [21] use the *graph pattern matching* model to formulate the VNF placement problem, without considering the VNF migration cost and the delay. Guo *et al.* [22] study throughput maximization in SDN-enabled networks with respect to service chaining specifications under various constraints, and do not consider the costs incurred by the VNFs. Ghaznavi *et al.* [23] propose an offline solution for an elastic virtual network function placement problem that minimizes the operational costs. Bari *et al.* [24] compute the required number and placement of VNFs to optimize the network operational costs and utilization using heuristic algorithms. Wang *et al.* [25] present VNF scaling models and online algorithms within one datacenter. Except [20], [21], [25], all other studies propose heuristic algorithms to solve the respective VNF placement problems, without giving any theoretical performance guarantee. On dynamic VNF scaling, similarly only heuristic approaches are discussed in a few system designs, *e.g.*, Stratos [15] and E2 [16]. Going substantially beyond the existing work, we aim to design an online algorithm for dynamic VNF deployment and scaling across datacenters, and provide solid theoretical guarantee of algorithm efficiency.

III. Problem Model

A. The NFV System

We consider a network service provider which rents virtual machines (VMs) from cloud datacenters distributed in different geographic locations, deploys virtualized network functions (VNFs) on the VMs, and assembles service chains to serve traffic flows between arbitrary sender and receiver locations. There are in total M types of VNF and I datacenters. Without loss of generality, the system works in a time slotted fashion within a large time span of $1, 2, \ldots, T$. Each time slot represents a decision interval, which is much longer than a typical end-to-end flow delay. Let $[X]$ represent the set $\{1, 2, \ldots, X\}$ throughout the paper, *e.g.*, $[M] = \{1, 2, \ldots, M\}$ is the set of different VNFs.

The VM instances running different VNFs are referred to as *VNF instances*. The flow processing capacities of VNF instances may differ, depending on resource capacities of the respective VMs, including CPU, memory and network I/O. We use $b_{m,i}$ to denote the processing capacity of each instance of VNF m in datacenter i, in terms of the maximum flow rate that it can process in each time slot without causing extraordinary processing and queueing delays. Instances of the same VNF are deployed on the same types of VMs in one datacenter, while VM types may differ from one datacenter to another, leading to different processing capacities of instances of the same VNF across different datacenters.

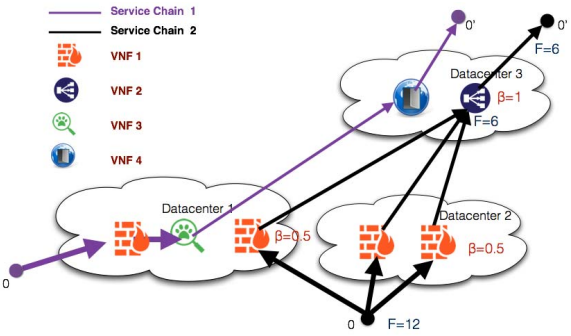

Fig. 1. An example of NFV service chains deployed over geo-distributed datacenters.

Up to K traffic flows may co-exist at any time in the system, each consisting of data packets from source s_k (location) to destination z_k (location), traversing the same service chain $p_k, k \in [K]$. A service chain p_k contains an ordered sequence of selected VNFs from set $[M]$, *i.e.*, $p_k \subseteq [M]$, and the hops in the service chain can be described by indicators $h_{k,m,m'}, \forall m, m' \in p_k$: $h_{k,m,m'} = 1$, if $m \rightarrow m'$ is a hop of p_k; and $h'_{k,m,m} = 0$ otherwise. For completeness of formulation, we add a dummy VNF 0 and a dummy VNF $0'$ to the head and the tail of each service chain, respectively, such that $h_{k,0,m} = 1$ if m is the first VNF in the chain and $h_{k,m,0'} = 1$ if m is the last VNF in the chain. A flow can follow different paths from the source to the destination, via different instances of the same VNF along its service chain. Fig. 1 shows two service chains deployed over multiple datacenters, where the flow along service chain 2 can be distributed to three instances of VNF 1 located in 2 different datacenters.

Let $F_k^{(t)}$ be the rate of flow k at the source, going into instances of the first VNF in its service chain at time t. Note that we capture dynamic flow arrivals and departures by allowing the flow rate to be zero if the flow does not exist at the time. Even in the same time slot, the flow rate may change at different hops of the VNF chain: some VNFs perform tunneling gateway functions (*e.g.*, IPSec/SSL VPN and media gateways), converting packets from one format to another, which may increase the packet size for encapsulation or decrease the packet size for decapsulation [26]; some VNFs perform security functions (*e.g.*, firewalls and intrusion detection), dropping packets which violate security policies [27]. We use $\beta_{k,m}$ to denote the average change ratio of flow rates of flow k on VNF m, such that the outgoing rate of flow k after traversing an instance of VNF m is on average $\beta_{k,m}$ times the incoming rate. In practice, we can estimate $\beta_{k,m}$ based on the past history and calibrate its value

over time. For ease of problem formulation, we define $\bar{\beta}_{k,m}$ as the cumulative rate change ratio of flow k before it goes through VNF m, which is the ratio of the overall rate of the flow coming into all instances of VNF m over the initial flow rate $F_k^{(t)}$. $\bar{\beta}_{k,m}$ can be computed using $\beta_{k,m}$'s as follows:

$$\bar{\beta}_{k,m_{k,s}} = 1, \text{ where } m_{k,s} \text{ is the first VNF in service chain } p_k,$$
$$\bar{\beta}_{k,m} = \sum_{m' \in p_k} h_{k,m',m} \bar{\beta}_{k,m'} \beta_{k,m'}, \quad \forall m \in p_k / \{m_{k,s}\}.$$

The total flow rate arriving at all instances of VNF m of flow k at t, $\hat{F}_{k,m}^{(t)}$, can be computed based on $F_k^{(t)}$ and $\bar{\beta}_{k,m}$ as

$$\hat{F}_{k,m}^{(t)} = F_k^{(t)} \bar{\beta}_{k,m} \tag{1}$$

For the example of service chain 2 in Fig. 1, suppose initial rate of the flow going into the service chain is $F_2^{(1)} = 12$; then we have $\hat{F}_{2,1}^{(1)} = F_2^{(1)} = 12$, $\hat{F}_{2,2}^{(1)} = 6$ (since $\beta_{2,1} = 0.5$) and $\hat{F}_{2,0'}^{(1)} = 6$ (since $\beta_{2,2} = 1$).

B. Decision Variables

We seek to make the following decisions on VNFs deployment and flow routing at each time slot $t \in [T]$: (i) $q_{m,i}^{(t)}$, the number of instances of VNF m to deploy in datacenter i, $\forall m \in [M], i \in [I]$. An instance of a VNF can be shared by multiple flows whose service chain includes this VNF. (ii) $x_{k,m,i,m',i'}^{(t)}$, the overall egress rate of flow k from instances of VNF m at datacenter i to instances of VNF m' at datacenter i', $\forall k \in [K], m, m' \in [M], i, i' \in [I]$. For ease of optimization formulation, we also introduce $y_{k,m,i}^{(t)}$ to denote the total ingress rate of flow k to instances of VNF m at datacenter i, *i.e.*, $y_{k,m,i}^{(t)} = \sum_{i' \in [I]/\{i\}} \sum_{m' \in [M]} h_{k,m',m} x_{k,m',i',m,i}^{(t)}$. Here $q_{m,i}^{(t)}$'s are non-negative integers, while $x_{k,m,i,m',i'}^{(t)}$'s and $y_{k,m,i}^{(t)}$'s are non-negative real numbers. In the example in Fig. 1, $q_{1,1}^{(1)} = 2$, $q_{1,2}^{(1)} = 2$ and $q_{2,3}^{(1)} = 1$; for service chain 2, supposing the incoming flow is evenly distributed to the three instances of VNF 1 in datacenters 1 and 2, then $x_{2,1,1,2,3}^{(1)} = 2$, $x_{2,1,2,2,3}^{(1)} = 4$ and $y_{2,2,3}^{(1)} = 6$.

C. Cost Structure

We aim to minimize the overall operational cost of the network service provider during $[T]$, with the following costs considered.

1) VNF Running Cost: The network service provider pays the cloud operator for renting VMs to run the VNF instances. Let $c_{m,i}^{(t)}$ be the cost of operating an instance of VNF m in datacenter i in time slot t. The overall cost for VM rentals in all datacenters in t is:

$$C_R^{(t)} = \sum_{i \in [I]} \sum_{m \in [M]} c_{m,i}^{(t)} q_{m,i}^{(t)} \tag{2}$$

2) VNF Deployment Cost: Launching a new VNF instance commonly involves transferring a VM image containing the network function to the hosting datacenter/server, and booting and attaching the image to device. The deployment cost is typically considered on the order of the cost to run a server for a number of seconds or minutes [28]. Let $\delta_{m,i}$ denote the cost for deploying an instance of VNF m in datacenter i. Let $\rho_{m,i}^{(t)}$ denote the number of new instances of VNF m to be deployed in datacenter i in t, which can be calculated as

$$\rho_{m,i}^{(t)} = \max\{0, q_{m,i}^{(t)} - q_{m,i}^{(t-1)}\} \tag{3}$$

The total deployment cost for all new VNF instances in all datacenters in t is computed as:

$$C_D^{(t)} = \sum_{i \in [I]} \sum_{m \in [M]} \delta_{m,i} \rho_{m,i}^{(t)} \tag{4}$$

Note that we consider VNF migration cost within the VNF deployment cost. In real-world systems, the VNF destruction cost is relatively small [29] and we treat the VNF deployment cost as the major cost incurred during VNF migration. If we are to include the VNF destruction cost, we may just set $\rho_{m,i}^{(t)} = |q_{m,i}^{(t)} - q_{m,i}^{(t-1)}|$ in (3).

3) Flow Transfer Cost: To transfer flows into and out of a datacenter, a bandwidth charge may be incurred (*e.g.*, [30]). Let d_i^{in} and d_i^{out} respectively be the costs of sending a unit of a flow into and out of datacenter i in a time slot. The overall bandwidth cost of all flows across all datacenters in t is:

$$\begin{aligned} C_T^{(t)} &= \sum_{k \in [K]} \sum_{m' \in [M]} \sum_{i' \in [I]/\{i\}} \sum_{m \in [M]} \sum_{i \in [I]} h_{k,m',m} d_i^{in} x_{k,m',i',m,i}^{(t)} \\ &\quad + \sum_{k \in [K]} \sum_{m \in [M]} \sum_{i \in [I]} \sum_{m' \in [M]} \sum_{i' \in [I]/\{i\}} h_{k,m,m'} d_i^{out} x_{k,m,i,m',i'}^{(t)} \end{aligned}$$

which is equivalent to

$$\begin{aligned} C_T^{(t)} &= \sum_{k \in [K]} \sum_{m \in [M]} \sum_{i \in [I]} (d_i^{in} + d_i^{out} \beta_{k,m}) y_{k,m,i}^{(t)} \\ &\quad - \sum_{k \in [K]} \sum_{m \in [M]} \sum_{m' \in [M]} \sum_{i \in [I]} h_{k,m,m'} (d_i^{in} + d_i^{out}) x_{k,m,i,m',i}^{(t)} \end{aligned}$$

where the deduction is to remove flows between VNF instances in the same datacenter from the cost computation.

4) End-to-End Flow Delay: As an important performance indicator, the end-to-end delay of each flow to pass through its service chain should be minimized, which is mainly decided by locations of the VNF instances that it traverses, *i.e.*, delays on the hops. Let $l_{i,j}$ denote the delay between node i and node j in the set of all datacenters, flow sources, and flow destinations, such that $l_{i,j} = l_{j,i}$ and $l_{i,i} = 0$. We use a generalized $\alpha-$relaxed triangle inequality [31] to describe delays between nodes:

$$|l_{a,b} - l_{b,c}| \leq \alpha l_{a,c}, \quad \forall a, b, c \in [I] \cup_{k \in [K]} \{s_k, z_k\} \tag{5}$$

If $\alpha \leq 1$, (5) becomes the normal triangle inequality, $l_{a,b} \leq l_{a,c} + l_{b,c}$; $\alpha > 1$ implies violation of the normal triangle inequality, which is common for the Internet delay space.

The average end-to-end delay of a flow k can be computed by summing up the following: (i) The average delay from source s_k to datacenters hosting the first VNF of service chain p_k, $\sum_{m \in [M]} h_{k,0,m} (\frac{\sum_{i \in [I]} l_{s_k,i} y_{k,m,i}^{(t)}}{F_k^{(t)}})$. The summation

over all datacenters represents that the flow can be dispatched to different instances of the first VNF m ($h_{k,0,m} = 1$) in different datacenters, and dividing the sum by $F_k^{(t)}$ removes the impact of flow rate, leaving only delay in the result. (ii) The average delay in each hop $m \rightarrow m'$ of service chain p_k ($h_{k,m,m'} = 1$), $h_{k,m,m'} \frac{\sum_{i\in[I]} \sum_{i'\in[I]} l_{i,i'} x^{(t)}_{k,m,i,m',i'}}{\beta_{k,m}\bar{\beta}_{k,m} F_k^{(t)}}$. The summation over all datacenter pairs models that both VNF m and VNF m' can be deployed in multiple datacenters, and dividing the sum by the overall flow rate at this hop, $\beta_{k,m}\bar{\beta}_{k,m}F_k^{(t)}$, removes the influence of flow rates in delay calculation. (iii) The average delay from the datacenters hosting instances of the last VNF of service chain p_k to destination d_k, $\sum_{m\in[M]} h_{k,m,0'}(\frac{\sum_{i\in[I]} l_{i,z_k}\beta_{k,m} y^{(t)}_{k,m,i}}{\beta_{k,m}\bar{\beta}_{k,m}F_k^{(t)}})$. Hence the average end-to-end delay of flow k in t is:

$$\sum_{m\in[M]}\sum_{i\in[I]} \frac{h_{k,0,m} l_{s_k,i}}{F_k^{(t)}} y^{(t)}_{k,m,i} + \sum_{m\in[M]}\sum_{i\in[I]} \frac{h_{k,m,0} l_{i,z_k}}{\bar{\beta}_{k,m} F_k^{(t)}} y^{(t)}_{k,m,i} + \sum_{m\in[M]}\sum_{i\in[I]}\sum_{m'\in[M]}\sum_{i'\in[I]} \frac{h_{k,m,m'} l_{i,i'}}{\beta_{k,m}\bar{\beta}_{k,m} F_k^{(t)}} x^{(t)}_{k,m,i,m',i'}$$

By multiplying the average end-to-end delay of each flow k with a weight parameter $a_k^{(t)}$, we convert the delay to a delay cost, to be aggregated with other costs for minimization in our optimal decision making. The overall delay cost of all flows in time slot t is:

$$C_E^{(t)} = \sum_{k\in[K]}\sum_{m\in[M]}\sum_{i\in[I]} \xi^{(t)}_{k,m,i} y^{(t)}_{k,m,i} + \sum_{k\in[K]}\sum_{m\in[M]}\sum_{i\in[I]}\sum_{m'\in[M]}\sum_{i'\in[I]} \omega^{(t)}_{k,m,i,m',i'} x^{(t)}_{k,m,i,m',i'} \tag{6}$$

where

$$\xi^{(t)}_{k,m,i} = \frac{a_k^{(t)} h_{k,0,m} l_{s_k,i}}{F_k^{(t)}} + \frac{a_k^{(t)} h_{k,m,0'} l_{i,z_k}}{\bar{\beta}_{k,m} F_k^{(t)}}$$

and

$$\omega^{(t)}_{k,m,i,m',i'} = \frac{a_k^{(t)} h_{k,m,m'} l_{i,i'}}{\beta_{k,m}\bar{\beta}_{k,m} F_k^{(t)}}.$$

Such a conversion of delay to a delay cost, in order to sum it up in the total cost for minimization, can be interpreted from the angle of multi-objective optimization [32]: the classic technique to treat an optimization problem where we have more than one objectives (*e.g.*, minimize operating cost and minimize delay as in our case), *i.e.*, a multi-objective optimization problem, is to convert it into a single objective optimization problem, where the new objective is a weighted sum of the original multiple objectives.

D. The Offline Cost Minimization Problem

Assuming full knowledge of the system in $[T]$, we can formulate an offline VNF deployment and flow routing problem in (7), for overall cost minimization. Important notation is listed in Table I for ease of reference.

TABLE I
KEY NOTATION

I	# of datacenters
M	# of VNF types
T	# of time slots
K	# of flows
s_k, z_k	source / destination of flow k
$F_k^{(t)}$	flow rate of flow k at time t
$\hat{F}_{k,m}$	total rate of flow k through VNF m
p_k	service chain that flow k follows
$a_k^{(t)}$	delay-to-cost conversion coefficient for flow k at t
$h_{k,m,m'}$	indicator of whether $m \rightarrow m'$ is a hop of p_k
$l_{i,j}$	delay from i to j
$\beta_{k,m}$	rate change ratio of flow k through VNF m
$\bar{\beta}_{k,m}$	accumulative rate change ratio of flow k before going through VNF m
$y^{(t)}_{k,m,i}$	total ingress flow rate of flow k to VNF m in datacenter i at t
$x^{(t)}_{k,m,i,m',i'}$	rate of flow k from VNF m in datacenter i to VNF m' in datacenter i' at t
$b_{m,i}$	processing capacity per instance of VNF m in datacenter i
$q^{(t)}_{m,i}$	# of instances of VNF m deployed in datacenter i in t
$\rho^{(t)}_{m,i}$	# of new instances of VNF m deployed in datacenter i in t
$\delta_{m,i}$	deployment cost for each new instance of VNF m in datacenter i
$c^{(t)}_{m,i}$	operational cost for running an instance of VNF m in datacenter i in t
d_i^{in}, d_i^{out}	flow transfer cost for transferring a unit of flow in one time slot in/out of datacenter i

P: minimize $\sum_{t\in[T]} C_R^{(t)} + C_D^{(t)} + C_T^{(t)} + C_E^{(t)}$ (7)

subject to:

$$\sum_{k\in[K]} y^{(t)}_{k,m,i} \le q^{(t)}_{m,i} b_{m,i}, \quad \forall t\in[T],\ i\in[I],\ m\in[M] \tag{8a}$$

$$\rho^{(t)}_{m,i} \ge q^{(t)}_{m,i} - q^{(t-1)}_{m,i}, \quad \forall t\in[T],\ i\in[I],\ m\in[M] \tag{8b}$$

$$\sum_{i\in[I]} y^{(t)}_{k,m,i} = \hat{F}^{(t)}_{k,m}, \quad \forall t\in[T],\ m\in[M],\ k\in[K] \tag{8c}$$

$$y^{(t)}_{k,m,i} = \sum_{i'\in[I]}\sum_{m'\in[M]} h_{k,m',m} x^{(t)}_{k,m',i',m,i} \quad \forall t\in[T],\ m\in[M]/\{m_{k,s}\},\ i\in[I],\ k\in[K] \tag{8d}$$

$$\beta_{k,m} y^{(t)}_{k,m,i} = \sum_{m'\in[M]}\sum_{i'\in[I]} h_{k,m,m'} x^{(t)}_{k,m,i,m',i'} \quad \forall t\in[T],\ m\in[M]/\{m_{k,z}\},\ i\in[I],\ k\in[K] \tag{8e}$$

$$x^{(t)}_{k,m,i,m',i'} \ge 0, \quad \forall t\in[T],\ m,m'\in[M],\ i,i'\in[I],\ k\in[K] \tag{8f}$$

$$y^{(t)}_{k,m,i} \ge 0, \quad \forall t\in[T],\ k\in[K],\ m\in[M],\ i\in[I] \tag{8g}$$

$$q^{(0)}_{m,i} = 0, \quad \forall m\in[M],\ i\in[I], \tag{8h}$$

$$q^{(t)}_{m,i} \in \{0,1,2,\ldots\}, \quad \forall t\in[T],\ m\in[M],\ i\in[I] \tag{8i}$$

$$\rho^{(t)}_{m,i} \in \{0,1,2,\ldots\}, \quad \forall t\in[T],\ m\in[M],\ i\in[I] \tag{8j}$$

Here $m_{k,s}$ ($m_{k,z}$) denotes the first (last) VNF in flow k's service chain. Constraint (8a) guarantees that the total

incoming flow rate to instances of VNF m in datacenter i does not exceed the processing capacity of the deployed instances. Constraints (8b) and (8j) are derived from the definition of auxiliary variable $\rho_{m,i}^{(t)}$ in (3). (8c) shows that the total incoming rate of flow k to VNF m in all datacenters should equal the (changed) aggregate rate of the flow at this hop, where $\hat{F}_{k,m}^{(t)}$ is decided by the initial flow rate and the change ratios along the service chain as in Eqn. (1). (8d) and (8e) guarantee flow conservation (considering flow rate change ratios $\beta_{k,m}$) for each flow at instances of each VNF in each datacenter, by connecting ingress and egress flows. In our model, the overall capacity limits are not considered in each datacenter, as the capacities provisioned are typically much larger than the demand from one service such as our VNF system.

Theorem 1: The offline optimization problem (7) is NP-hard.

Proof: The offline problem (7) is a mixed integer linear programming (i.e., MILP), and can be reduced from the *minimum (multiple) knapsack (covering) problem* (MKP) [33], which is known to be NP-hard. In MKP, a set of items are given, each with a weight and a value; the objective is to find a minimum-weight subset of items (each item can have multiple copies) such that the total value of the items in the subset meets some specified demand. We reduce MKP to (7) as follows: We set $T = 1$, and if we only consider the VNF running cost and treat other costs as 0 (i.e., $C_D^{(t)} = C_T^{(t)} = C_E^{(t)} = 0$, when the coefficients of these costs are 0), then the value for each item is $b_{m,i}$, the weight is $c_{m,i}^{(t)}$, $C_R^{(t)}$ presents the total weight, and the required demand in the MKP is $\sum_{k\in[K]} y_{k,m,i}^{(t)}$, which is satisfied by constraint (8a). Therefore our offline problem (7) is at least as difficult as the MKP. □

In an online setting, all parameters, variables and constraints related to time slot t are revealed upon the arrival of the corresponding time. Due to constraints (8b), decisions of one time slot are coupled with those in another. We seek to design an efficient online algorithm that produces VNF deployment and flow routing decisions based only on the current flow information and past history. Our offline problem provides us a better understanding of the problem at hand to facilitate our online algorithm design, and also presents a performance benchmark for our online algorithm: we will show that our online algorithm can achieve a good competitive ratio, computed by dividing the overall cost it achieves by the offline minimum cost derived by solving (7) exactly.

Our design of the online algorithm is divided into two steps. First, we relax the in tegrality constraints (8i) and (8j) in (7), obtain the fractional offline optimization problem P_f as follows, and apply a novel regularization method to design an online algorithm for solving this relaxed linear program (LP) in Sec. IV. We then design a dependent rounding algorithm to round the fractional solutions to feasible solutions of MILP (7) in Sec. V.

$$\mathbf{P_f}\text{: minimize } \sum_{t\in[T]} C_R^{(t)} + C_D^{(t)} + C_T^{(t)} + C_E^{(t)}$$
$$\text{subject to: constraints (8a) to (8h)}$$
$$q_{m,i}^{(t)} \geq 0, \quad \forall t \in [T],\ m \in [M],\ i \in [I] \quad (9i)$$
$$\rho_{m,i}^{(t)} \geq 0, \quad \forall t \in [T],\ m \in [M],\ i \in [I] \quad (9)$$

IV. The Fractional Online Algorithm via Regularization

Regularization

The key idea of our online algorithm to solve P_f is to lift constraint (8b) to the objective function using a smooth convex function. By doing this, decisions in time slots $t-1$ and t can be decoupled and the resulting relaxed offline problem, denoted by $\tilde{P}_f$, can be readily decomposed into a set of subproblems, each to be efficiently solved in one time slot.

To lift (8b) into the objective function in (7), we observe that (8b) originates from the definition of variable $\rho_{m,i}^{(t)}$ in (3), the number of new instances of VNF m to be deployed in datacenter i in t. $\rho_{m,i}^{(t)}$ is involved in the deployment cost $C_D^{(t)}$ as defined in (4), which is the second term of the objective function in (7). We substitute $\rho_{m,i}^{(t)}$ in (7) by a function of $q_{m,i}^{(t-1)}$ and $q_{m,i}^{(t)}$ via the regularization technique from the online learning literature [34], removing constraint (8b). In particular, we use the following relative entropy function, a commonly adopted convex regularizer, as the basis of the function to approximate $\rho_{m,i}^{(t)}$ in (3):

$$\Delta(q_{m,i}^{(t)} || q_{m,i}^{(t-1)}) = q_{m,i}^{(t)} \ln \frac{q_{m,i}^{(t)}}{q_{m,i}^{(t-1)}} + q_{m,i}^{(t-1)} - q_{m,i}^{(t)} \quad (10)$$

This relative entropy function is obtained by summing the relative entropy ($q_{m,i}^{(t)} \ln \frac{q_{m,i}^{(t)}}{q_{m,i}^{(t-1)}}$) and a linear term representing the movement cost ($q_{m,i}^{(t-1)} - q_{m,i}^{(t)}$). Such a function, proven convex, is a widely adopted regularizer in online learning problems that involve l_1-norm constraints [6], [34], such as in our case (constraint (8b)). Further, to approximate $\rho_{m,i}^{(t)}$ in (3), we add a constant term $\frac{\epsilon}{MI}$ to both $q_{m,i}^{(t)}$ and $q_{m,i}^{(t-1)}$ in the relative entropy term in (10), where ϵ is a small positive constant, to ensure that the fraction is still valid when the number of VNF instances deployed in $t-1$ is zero, *i.e.*, $q_{m,i}^{(t-1)} = 0$. We also define a parameter $\eta = \ln(1 + \frac{MI}{\epsilon})$ and multiple the improved relative entropy function by $\frac{1}{\eta}$, which is related to our competitive ratio, to normalize the deployment cost by regularization.

We replace $\rho_{m,i}^{(t)}$ in (7) by the constructed function, and obtain $\tilde{P}_f$ as follows. Here we only present the subproblem at t, $\tilde{P}_f^{(t)}$, that $\tilde{P}_f$ is decomposed into, $\forall t \in [T]$. $\tilde{P}_f$ can be readily obtained by adding a summation over $t \in [T]$ in the objective function, *i.e.*, $\tilde{P}_f = \sum_{t\in[T]} \tilde{P}_f^{(t)}$, and repeating each constraint for each $t \in [T]$.

$$\tilde{\mathbf{P}}_\mathbf{f}^{(\mathbf{t})}\text{: minimize } C_R^{(t)} + C_T^{(t)} + C_E^{(t)} + \sum_{m\in[M]} \sum_{i\in[I]} \frac{\delta_{m,i}}{\eta} \times \left((q_{m,i}^{(t)} + \frac{\epsilon}{MI}) \ln \frac{q_{m,i}^{(t)} + \frac{\epsilon}{MI}}{q_{m,i}^{(t-1)} + \frac{\epsilon}{MI}} + q_{m,i}^{(t-1)} - q_{m,i}^{(t)} \right)$$
$$\text{subject to: constraints (8a)(8c)-(8h), (9i), for } t \quad (11)$$

Algorithm 1 An Online Regularization-Based Fractional Algorithm - *ORFA*

Input: $K, M, I, \boldsymbol{\beta}, \boldsymbol{\delta}, \mathbf{h}, \mathbf{b}, \mathbf{s}, \mathbf{z}, \mathbf{l}, \mathbf{d}^{in}, \mathbf{d}^{out}, \epsilon$
Output: $\mathbf{q}, \mathbf{x}, \mathbf{y}$
1 Initialization: $\mathbf{q} = \mathbf{0}, \mathbf{x} = \mathbf{0}, \mathbf{y} = \mathbf{0}$;
2 **for** *each time slot* $t \in [T]$ **do**
3 observe values of $F_k^{(t)}, a_k^{(t)}, c_{m,i}^{(t)}$, $\forall k \in [K], m \in [M], i \in [I]$;
4 compute $\hat{F}_k^{(t)}, \forall k \in [K]$, according to (1);
5 use the interior point method to solve $\tilde{P}_f^{(t)}$ in (11);
6 **return** optimal solutions $\mathbf{q}^{(t)}, \mathbf{x}^{(t)}, \mathbf{y}^{(t)}$;
7 **end**

Online Algorithm

Our online algorithm to derive a solution to P_f in (9), given in Alg. 1, runs by solving subproblem $\tilde{P}_f^{(t)}$ in (11) in each time slot t. Note that $q_{m,i}^{(t-1)}, \forall i \in [I], m \in [M]$, have been obtained when solving $\tilde{P}_f^{(t-1)}$ in time $t-1$, and their values are given as input to $\tilde{P}_f^{(t)}$ at t. Since $\tilde{P}_f^{(t)}$ is a convex optimization problem with linear constraints, it can be optimally solved in polynomial time, *e.g.*, using the interior point method [32]. It is obvious that the optimal solution of $\tilde{P}_f^{(t)}, \forall t \in [T]$, constitute a feasible solution to P_f (values of $\rho_{m,i}^{(t)}$'s can be easily set based on (3)), as given in the following theorem.

Theorem 2: The online fractional algorithm ORFA *produces a feasible solution of P_f in polynomial time.*

Competitive Ratio

$q_{m,i}^{(t)}$'s derived by *ORFA* are potentially fractional. We next show that the fractional solution can achieve a good competitive ratio in overall cost. In Sec. V, we will round the fractional solutions to integers, as well as show the final competitive ratio based on the ratio here plus the efficiency loss due to rounding.

Theorem 3: The overall cost in (7) that the solution derived by ORFA *achieves is at most* $\log(1 + MI/\epsilon) + 1 + 1/\phi$ *times the offline minimum overall cost derived by solving P in (7) exactly, where* $\phi = \min_{t \in [T], i \in [I], m \in [M]} \{q_{m,i}^{(t)} : q_{m,i}^{(t)} > 0\}$ *is the minimum positive number of instances deployed for any VNF, at any time slot and in any datacenter.*

The detailed proof is given in the supplementary materials.

We next give a bound on the value of the objective function achieved by any integer solution of (11), which will be used in our analysis of the final competitive ratio in the next section.

Theorem 4: The objective value of $\tilde{P}_f$ in (11) achieved by the best integral solution of (11), denoted by $\tilde{P}_I$, is at most $\log(1 + MI/\epsilon) + 2$ *times the offline minimum overall cost.*

Theorem 4 follows immediately Theorem 3, since the smallest integer value of ϕ in the ratio given in Theorem 3 is 1, denoting the minimum positive integral number of instances deployed for any VNF in any datacenter at any time.

V. An Online Dependent Rounding Scheme

A. The Dependent Rounding Scheme

The ORFA algorithm in Sec. IV computes a fractional solution $(\mathbf{q}^{(t)}, \mathbf{x}^{(t)}, \mathbf{y}^{(t)}, \boldsymbol{\rho}^{(t)})$, for the optimization problem in (9). However, the number of instances of each VNF to deploy should be integral, as captured by (8i) and (8j) in the offline problem. We need to round the fractional solution $\mathbf{q}^{(t)}$ to an integer solution $\bar{\mathbf{q}}^{(t)}$. Due to constraints (8a) - (8e), modifying $\mathbf{q}^{(t)}$ requires correspondingly modifying $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}, \boldsymbol{\rho}^{(t)})$ to maintain solution feasibility. Our goal in this section is to compute a rounded solution $(\bar{\mathbf{q}}^{(t)}, \bar{\mathbf{x}}^{(t)}, \bar{\mathbf{y}}^{(t)}, \bar{\rho}^{(t)})$ from $(\mathbf{q}^{(t)}, \mathbf{x}^{(t)}, \mathbf{y}^{(t)}, \boldsymbol{\rho}^{(t)})$ such that the rounded solution is feasible to (7).

We first note that a straightforward independent rounding scheme, where each variable is rounded up or down independently of other variables, may violate feasibility. For example, there is a chance that $q_{m,i}^{(t)}, \forall m \in [M], i \in [I]$, are rounded down, making it impossible to obtain a feasible routing solution. We therefore design a dependent rounding scheme that can exploit the inherent dependence of the variables in capacity constraint (8a) and flow conservation constraints (8d) and (8e). The key idea is that rounded-down VNF instance numbers will be compensated by rounded-up VNF instance numbers, as well as extra VNF instances deployed, guaranteeing a feasible $\bar{\mathbf{y}}^{(t)}$ and a feasible $\bar{\mathbf{x}}^{(t)}$ that satisfy (8a), (8d) and (8e).

Our dependent rounding scheme contains four modules: (i) a local clustering algorithm that separates the geo-distributed datacenter network into several clusters, each with at least two datacenters; (ii) an initialization module for weighted dependent rounding, that constructs a weighted star graph in each time slot; (iii) a weighted dependent rounding algorithm based on the clusters and start graph above to compute the rounded solution of $\bar{\mathbf{q}}^{(t)}$ and $\bar{\rho}^{(t)}$; (iv) a flow direction algorithm to find the feasible solution of $\bar{\mathbf{y}}^{(t)}$ and $\bar{\mathbf{x}}^{(t)}$, which will provide $(\bar{\mathbf{q}}^{(t)}, \bar{\mathbf{x}}^{(t)}, \bar{\mathbf{y}}^{(t)}, \bar{\rho}^{(t)})$ as the final feasible solution to (7).

1) Datacenter Clustering: Geo-distributed datacenters in practice often form natural clusters based on location proximity. Amazon EC2 datacenters [35] and Google datacenters [36] can be separated into 3 main clusters by continents: America, Europe and Asia datacenter groups. Delays between datacenters within the same cluster (intra-cluster delays) are substantially smaller than those between datacenters across different clusters (inter-cluster delays). We design a local clustering algorithm, as given in Alg. 2, to group datacenters in the system into clusters with intra-cluster delays no larger than R. Here R is defined to be the median of all the inter-datacenter delays (line 2). In Alg. 2, we construct the clusters by repeatedly merging existing clusters whose inter-cluster delays are no larger than R. Then we identify the isolated points (clusters including only one datacenter) and merge them to nearest clusters. The clustering algorithm is carried out once at the beginning of the system. By carrying out dependent rounding on top of the cluster structure, we will be able to bound the delay cost part in the overall cost in our competitive analysis, where the bound is related to R.

2) Star Graph Construction: In each time slot, we construct a star graph for each datacenter cluster (Alg. 3). We identify a buffer datacenter, $j_m^{(t)*}$ for each type of VNF m in each datacenter cluster, which has the smallest VNF running cost per unit flow processing capacity (line 4). We may deploy

Algorithm 2 The Local Clustering Algorithm

Input: $l_{i,i'}, \forall i, i' \in [I]$
Output: clusters of datacenters $\odot$

Treat each datacenter i as a singleton cluster;
$R = median_{i,i' \in [I]} l_{i,i'}$;
while *there are more clusters to merge* **do**
  for *Any two clusters* $\odot_u \neq \odot_v$ **do**
    if $l_{i,j} \leq R \ \forall i \in \odot_u, j \in \odot_v$ **then**
      Merge $\odot_u$ and $\odot_v$ into a new cluster $\odot_{u'}$.
    end
  end
end
for *any cluster* $\odot_u$ *with only one datacenter* i **do**
  Find the cluster $\odot_v$ with minimal value of $max_{j \in \odot_v} l_{i,j}$, merge $\odot_u$ into $\odot_v$.
end

Algorithm 3 The Initialization Module for Weighted Dependent Rounding in t, *INIT*

Input: $K, M, I, \mathbf{q^{(t)}}, \mathbf{c}, \mathbf{b}$
Output: $\mathbf{w^{(t)}}, \mathbf{p^{(t)}}, \mathbf{A}, \mathbf{B}, \boldsymbol{\Upsilon}, G(V_m^{(t)}, E_m^{(t)}), \forall m \in [M]$

1 **for** *each type of VNF* $m \in [M]$ **do**
2 Set $V_m^{(t)} = [I], E_m^{(t)} = \emptyset$;
3 **for** *each datacenter cluster* **do**
4 Decide datacenter $j_m^{(t)*}$ with minimal $\frac{c_{m,i}^{(t)}}{b_{m,i}}$; (if there is a tie, choose any one from the tied candidates)
5 Label this datacenter cluster as $\odot_{m,j_m^{(t)*}}$;
6 Set $A_{m,j_m^{(t)*}} = \odot_{m,j_m^{(t)*}} / \{j_m^{(t)*}\}$, $B_{m,j_m^{(t)*}} = \{j_m^{(t)*}\}$, $\Upsilon_{m,j_m^{(t)*}} = \{(i, j_m^{(t)*})\} \quad \forall i \in A_{m,j_m^{(t)*}}$;
7 **for** *each datacenter* $i \in A_{m,j_m^{(t)*}}$ **do**
8 **if** $q_{m,i}^{(t)} = \lfloor q_{m,i}^{(t)} \rfloor$ **then**
9 $\Upsilon_{m,j_m^{(t)*}} = \Upsilon_{m,j_m^{(t)*}} / \{(i, j_m^{(t)*})\}$;
10 **end**
11 **else**
12 Add the edge $(i, j_m^{(t)*})$ to $E_m^{(t)}$;
13 Associate a probability coefficient
$$p_{m,i}^{(t)} = \lfloor q_{m,i}^{(t)} \rfloor + 1 - q_{m,i}^{(t)} \qquad (12)$$
with the edge $(i, j_m^{(t)*})$;
14 Associate a weight coefficient
$$w_{m,i}^{(t)} = (q_{m,i}^{(t)} - \lfloor q_{m,i}^{(t)} \rfloor) \frac{b_{m,i}}{b_{m,j_m^{(t)*}}} \qquad (13)$$
with the edge $(i, j_m^{(t)*})$;
15 **end**
16 **end**
17 **end**
18 **end**

extra instances of VNF m in such a buffer datacenter to absorb unserved flow due to round-down of VNF instance numbers

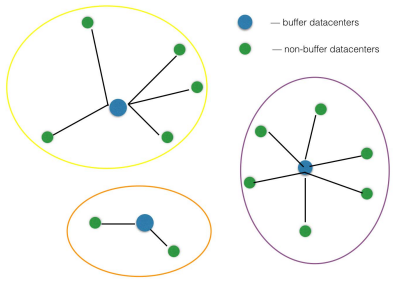

Fig. 2. Datacenter graph with disjoint stars.

in other non-buffer datacenters in the cluster. The buffer datacenter is the centre of the star graph for the cluster. Let $A_{m,j_m^{(t)*}}$ denote the non-buffer datacenters in cluster $\odot_{m,j_m^{(t)*}}$, $B_{m,j_m^{(t)*}}$ denote the buffer datacenter in $\odot_{m,j_m^{(t)*}}$, and $\Upsilon_{m,j_m^{(t)*}}$ denote the set of fractional edges in $\odot_{m,j_m^{(t)*}}$ with $0 < p_{m,i}^{(t)} < 1$ (line 6). For each datacenter in each cluster except the buffer (line 7), if the computed instance number of VNF m by the online algorithm ORFA is already an integer, we ignore this datacenter in the rounding scheme and remove it from Υ (lines 8-9); otherwise, we add an edge connecting the datacenter to the buffer datacenter, and associate a probability $p_{m,i}^{(t)}$ and a weight with the edge $w_{m,i}^{(t)}$ (lines 13-14). $p_{m,i}^{(t)}$ is computed by (12) and will be used for rounding. $w_{m,i}^{(t)}$ is computed by (13), and will be used for determining the number of instances of VNF m to be deployed in the buffer datacenter. After running this initialization module, the data center graph is dissected into disjoint stars, as illustrated in Fig. 2

3) Weighted Dependent Rounding: We are now ready to present the Online Weighted Dependent Rounding (OWDR) algorithm in Alg. 4, which computes a feasible integer solution of $\bar{q}_{m,i}^{(t)}$ in each time slot t. For each type of VNF m, OWDR runs a series of rounding iterations for each cluster (line 1). For each cluster $\odot_{m,j_m^{(t)*}}$, we find a maximum path in its star topology (of length at most 2) that will be the path from a datacenter in $A_{m,j_m^{(t)*}}$ to the buffer datacenter $j_m^{(t)*}$ to another datacenter in $A_{m,j_m^{(t)*}}$. Then the two edges are placed into two distinct matchings (line 4). If only one floating edge remains (this can happen only in the last iteration), we choose this edge as the maximal path with length 1. During each iteration, we let the probability of one of these two edges, $p_{m,i_1}^{(t)}$ or $p_{m,i_2}^{(t)}$ round to 0 or 1, decided by the coupled coefficient κ_1 and κ_2, which will decrease the number of elements in $\Upsilon_{m,j_m^{(t)*}}$ by 1. Finally, the solution $\bar{q}_{m,i}^{(t)}$ can be produced by the original integral part $\lfloor q_{m,i}^{(t)} \rfloor$ plus the rounded fractional part $p_{m,i}^{(t)}$. Furthermore, based on the integral solutions $\bar{q}_{m,i}^{(t)}$, we can then calculate $\bar{\rho}_{m,i}^{(t)}$ as $\max\{0, \bar{q}_{m,i}^{(t)} - \bar{q}_{m,i}^{(t-1)}\}$ (line 12).

The following theorem shows that the numbers of instances of each VNF m to deploy in the datacenters, $\bar{q}_{m,i}^{(t)}$ produced by OWDR, are integers and are sufficient to serve all the incoming flows that need this VNF at t.

Theorem 5: Elements of $\bar{\mathbf{q}}^{(t)}$ *produced by OWDR are non-negative integers and satisfy* $\sum_{i \in [I]} q_{m,i}^{(t)} b_{m,i} \geq \sum_{k \in [K]} \hat{F}_{k,m}^{(t)}, \forall m \in [M], t \in [T]$.

Algorithm 4 The Online Weighed Dependent Rounding Algorithm (OWDR) in t

Input: $\mathbf{q^{(t)}}$,star graphs $G(V_m^{(t)}, E_m^{(t)}), \forall m \in [M]$, $\mathbf{p^{(t)}}, \mathbf{w^{(t)}}, \bar{\mathbf{q}}^{\mathbf{(t-1)}}$
Output: $\bar{\mathbf{q}}^{(\mathbf{t})}, \bar{\rho}^{(t)}$

for *each type of VNF* $m \in [M]$ **do**
  for *each datacenter cluster* $\odot_{m,j_m^{(t)*}}$ **do**
    while $\Upsilon_{m,j_m^{(t)*}} \neq \emptyset$ **do**
      Run the depth-first-search (DFS) algorithm to obtain maximal path $\mathbb{P}$ in the star graph (A, B, Υ) and partition the edge set of $\mathbb{P}$ into two matchings Ψ_1 and Ψ_2 ;
      Define

$$\kappa_1 = \min\{\gamma > 0 | (\exists (i_1, j_m^{(t)*}) \in \Psi_1 : p_{m,i_1}^{(t)} + \gamma = 1) \vee (\exists (i_2, j_m^{(t)*}) \in \Psi_2 : p_{m,i_2}^{(t)} - \frac{w_{m,i_1}^{(t)}}{w_{m,i_2}^{(t)}}\gamma = 0)\}$$

$$\kappa_2 = \min\{\gamma > 0 | (\exists i_1, j_m^{(t)*}) \in \Psi_1 : p_{m,i_1}^{(t)} - \gamma = 0) \vee (\exists i_2, j_m^{(t)*}) \in \Psi_2 : p_{m,i_2}^{(t)} + \frac{w_{m,i_2}^{(t)}}{w_{m,i_1}^{(t)}}\gamma = 1)\}$$

      With probability $\frac{\kappa_2}{\kappa_1+\kappa_2}$, set

$$p_{m,i_1}^{(t)} = p_{m,i_1}^{(t)} + \kappa_1, \quad \forall (i_1, j_m^{(t)*}) \in \Psi_1$$

$$p_{m,i_2}^{(t)} = p_{m,i_2}^{(t)} - \kappa_1 \frac{w_{m,i_1}^{(t)}}{w_{m,i_2}^{(t)}}, \quad \forall (i_2, j_m^{(t)*}) \in \Psi_2$$

      Remove $(i_1, j_m^{(t)*})$ or $(i_2, j_m^{(t)*})$ from $\Upsilon_{m,j_m^{(t)*}}$ if $p_{m,i_1}^{(t)} \in \{0,1\}$ or $p_{m,i_2}^{(t)} \in \{0,1\}$
      With probability $\frac{\kappa_1}{\kappa_1+\kappa_2}$, set

$$p_{m,i_1}^{(t)} = p_{m,i_1}^{(t)} - \kappa_2, \quad \forall (i_1, j_m^{(t)*}) \in \Psi_1$$

$$p_{m,i_2}^{(t)} = p_{m,i_2}^{(t)} + \kappa_2 \frac{w_{m,i_1}^{(t)}}{w_{m,i_2}^{(t)}}, \quad \forall (i_2, j_m^{(t)*}) \in \Psi_2$$

      Remove $(i_1, j_m^{(t)*})$ or $(i_2, j_m^{(t)*})$ from $\Upsilon_{m,j_m^{(t)*}}$ if $p_{m,i_1}^{(t)} \in \{0,1\}$ or $p_{m,i_2}^{(t)} \in \{0,1\}$
    end
  end
  for *each datacenter* $i \in A$ **do**

$$\bar{q}_{m,i}^{(t)} = \lfloor q_{m,i}^{(t)} \rfloor + p_{m,i}^{(t)}$$
$$\bar{\rho}_{m,i}^{(t)} = \max\{0, \bar{q}_{m,i}^{(t)} - \bar{q}_{m,i}^{(t-1)}\}$$

  end
end

The detailed proofs of the lemmas are given in the supplementary materials.

4) Flow Redirection: $\bar{\mathbf{q}}^{(t)}$ and $\bar{\rho}^{(t)}$ produced by OWDR together with flow routing decisions $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$ from Alg. 1 may well not be a feasible solution of (7). We next obtain $\bar{\mathbf{x}}^{(t)}$ and $\bar{\mathbf{y}}^{(t)}$ based on $(\bar{\mathbf{q}}^{(t)}, \bar{\rho}^{(t)})$ that leads to a complete feasible solution of (7), by solving a linear optimization problem as follows. The LP in (14) is derived from (7) by plugging in the values of $\bar{\mathbf{q}}^{(t)}$ and $\bar{\rho}^{(t)}$ obtained by OWDR and removing the resulting fixed *VNF running cost* and *VNF deployment cost* from the objective and the relevant constraints.

$$\begin{aligned} &\text{minimize } C_T^{(t)} + C_E^{(t)} \\ &\text{subject to: constraints (8a)(8c)-(8g)} \end{aligned} \tag{14}$$

The above LP can be exactly solved using the interior point method in polynomial time.

B. The Complete Online Algorithm

We summarize our complete online algorithm in Alg. 5, which incorporates the online regularization-based algorithm (i.e., ORFA) to get an online fractional solution, together with the dependent rounding algorithm (i.e., OWDR). Here $ORFA^{(t)}$ is referring to the steps executed in time slot t of the online algorithm ORFA, *i.e.*, lines 3-6 in Alg. 1. $INIT$ is the routine in Alg. 3 and $OWDR$ is the routine in Alg. 4.

Algorithm 5 The Complete Online Algorithm - *COA*

Input: $K, M, I, \boldsymbol{\beta}, \boldsymbol{\delta}, \mathbf{h}, \mathbf{b}, \mathbf{s}, \mathbf{z}, \mathbf{l}, \mathbf{d}^{in}, \mathbf{d}^{out}, \epsilon$
Output: $\bar{\mathbf{q}}, \bar{\rho}, \bar{\mathbf{x}}, \bar{\mathbf{y}}$

1 Initialization: $\mathbf{q} = \mathbf{0}, \mathbf{x} = \mathbf{0}, \mathbf{y} = \mathbf{0}, \bar{\mathbf{q}} = \mathbf{0}$;
2 call the local clustering algorithm in Alg. 2;
3 **for** *each time slot* $t \in [T]$ **do**
4 $(\mathbf{q}^{(t)}, \mathbf{x}^{(t)}, \mathbf{y}^{(t)}) = ORFA^{(t)}(K, M, I, \boldsymbol{\beta}, \boldsymbol{\delta}, \mathbf{h}, \mathbf{b}, \mathbf{s}, \mathbf{z}, \mathbf{l}, \mathbf{d}^{in}, \mathbf{d}^{out}, \epsilon)$;
5 $(\mathbf{w^{(t)}}, \mathbf{p^{(t)}}, \mathbf{A}, \mathbf{B}, \boldsymbol{\Upsilon}, G(V_m^{(t)}, E_m^{(t)}), \forall m \in [M]) = INIT(K, M, I, \mathbf{q^{(t)}}, \mathbf{c}, \mathbf{b})$;
6 $(\bar{\mathbf{q}}^{(t)}, \bar{\rho}^{(t)}) = OWDR(\mathbf{q}^{(t)}, G(V_m^{(t)}, E_m^{(t)}), \forall m \in [M], \mathbf{p^{(t)}}, \mathbf{w^{(t)}}, \bar{\mathbf{q}}^{(t-1)})$;
7 Obtain $\bar{\mathbf{x}}^{(t)}$ and $\bar{\mathbf{y}}^{(t)}$ by solving LP (14) using interior point method;
8 **end**

Theorem 6: The complete online algorithm COA runs in polynomial time.

Theorem 7: The complete online algorithm COA in Alg. 5 computes a feasible solution $(\bar{\mathbf{q}}^{(t)}, \bar{\mathbf{x}}^{(t)}, \bar{\mathbf{y}}^{(t)}, \bar{\rho}^{(t)})$ *of the original problem (7).*

Proof: Since $\bar{\mathbf{q}}^{(t)}$ satisfies the condition in Theorem 5, we can find a feasible solution of $\mathbf{y}^{(t)}$ which satisfies constraints (8a) and (8c), and then a feasible solution of $\mathbf{x}^{(t)}$ satisfying the flow conservation constraints (8d) and (8e). That is, we can find a feasible solution when solving LP (14), and $(\bar{\mathbf{q}}^{(t)}, \bar{\mathbf{x}}^{(t)}, \bar{\mathbf{y}}^{(t)}, \bar{\rho}^{(t)})$ is feasible for the original problem (7). □

Competitive Analysis

We now analyze the competitive ratio of COA. Let $P_I(ORFA)$ denote the objective value of problem (11) achieved by its best integer solution. Let $\bar{P}_I(COA)$ be the

objective value of the original problem (7) achieved by the solution produced by COA, in which the VNF running cost, VNF deployment cost, flow transfer cost and delay cost are $\bar{P}_I(C_R)$, $\bar{P}_I(C_D)$, $\bar{P}_I(C_T)$ and $\bar{P}_I(C_E)$, respectively:

$$\bar{P}_I(COA) = \bar{P}_I(C_R) + \bar{P}_I(C_D) + \bar{P}_I(C_E) + \bar{P}_I(C_T)$$

The competitive ratio achieved by our complete online algorithm in Alg. 5 is computed as the worst-case ratio of $\bar{P}_I(COA)$ over the minimium overall cost computed by solving the offline problem (7) exactly.

Lemma 1: $\bar{P}_I(C_R)$ *is at most* 2 *times* $P_I(ORFA)$.

Lemma 2: $\bar{P}_I(C_D)$ *is no larger than* ϕ_1 *times* $P_I(ORFA)$, *where* $\phi_1 = \max_{t\in[T],m\in[M],i\in[I]} \frac{\delta_{m,i}}{c^{(t)}_{m,i}}$ *is the maximal ratio of deployment cost to operational cost per VNF instance.*

Lemma 3: $\bar{P}_I(C_T)$ *is no larger than* ϕ_2 *times* $P_I(ORFA)$, *where* $\phi_2 = \max_{t\in[T],m\in[M],i\in[I]} \frac{(d^{in}_i + d^{out}_i).b_{m,i}}{c^{(t)}_{m,i}}$ *is the maximal ratio of flow transfer cost to operational cost per VNF instance.*

Lemma 4: $\bar{P}_I(C_E)$ *is no larger than* ϕ_3 *times* $P_I(ORFA)$, *where*

$$\phi_3 = \max_{t\in[T],m\in[M],i\in[I],k\in[K],u,v\in S} \max\{\frac{\alpha l_{u,v} a^{(t)}_k b_{m,i}}{Rc^{(t)}_{m,i}}, \},$$
$$S = \cup_{k\in[K]}\{s_k, z_k\} \cup [I],$$

and α *is the coefficient for the relaxed triangle inequality defined in (5).*

The detailed proofs of the lemmas are given in the supplementary materials.

Theorem 8: (Final Competitive Ratio) $\bar{P}_I(COA)$ *is no larger than* $[\log(1 + MI/\epsilon) + 2](2 + \phi_1 + \phi_2 + \phi_3)$ *times the offline minimum objective value* P^*_I *of the original problem (7).*

Proof: According to Lemmas 1 – 4 and Theorem 4, we have

$$\begin{aligned}\bar{P}_I(COA) &\le (2 + \phi_1 + \phi_2 + \phi_3)P_I(OFRA) \\ &\le [\log(1 + MI/\epsilon) + 2](2 + \phi_1 + \phi_2 + \phi_3)P^*_I\end{aligned}$$

□

VI. Performance Evaluation

A. Simulation Setup

We use the topology of the physical network of Cogent to create the datacenter network [37]. The default number of datacenters is 50. Delays between datacenters are set proportional to their geographic distances, perturbed by multiplying a random number in $[0.8, 1.2]$. The flows are generated according to real web traffic based on the Wikipedia trace [38], which contains 20.6 billion HTTP requests within a 10-month period. We generate the geolocations of the sources and destinations of the flows uniformly at random over the Cogent datacenter network. We further boost the daily peak in the WikiPedia traffic to flash crowds generated using the model in [39]: R_{flash} = shock level × R_{normal}, where the *shock level* is the order of magnitude increase in the average request rate. Fig. 3 shows the input traffic rate to service chains, produced at shock level of 5 (default). Since the IP addresses in the traces are hidden due to privacy issues, we use the demographic density data of the global Internet users [40] to model the geo-distribution of the resource/destination pairs. A location with high demographic density is selected with a high probability to be a flow source or destination.

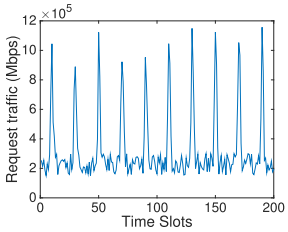

Fig. 3. Request traffic with flash crowds.

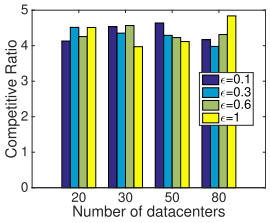

Fig. 4. Performance of ORFA with different # of datacenters.

We simulate the VNFs in the following table, with respective processing capacity, instance type (following those of Amazon EC2), and flow rate change ratios. The running cost, deployment cost and flow transfer cost for each VNF are set according to EC2's pricing of the respective instance type [30]. We set $a^{(t)}_k$'s for converting delay to cost to 1. We create 30 service chains, each containing 2∼5 VNFs randomly chosen from the four. Each time slot in our experiments corresponds to one hour.

B. Performance of ORFA

We first examine the ratio achieved by ORFA in Alg. 1 without rounding, by dividing the overall cost achieved by the fractional solution of ORFA by the offline minimum overall cost. The offline minimal cost is obtained by solving (7) exactly using CVX and MOSEK Optimizer. Due to the complexity of solving the offline problem with a large number of variables, we set the default number of time slots to be $T = 200$.

Fig. 4 shows that the number of datacenters does not influence the result significantly. Under our setting, datacenters are residing in locations with high Internet user density with higher probability according to the reality. Fig. 5 further shows that the performance is quite stable with the increase of the total number of time slots that the system spans. The impact of ϵ is not obvious as shown in Fig. 4 and Fig. 5. The theoretical worst case ratio given in Theorem 3 is larger when the number of datacenters is larger and smaller when ϵ is larger; here we observe that in practice, their impact is not obvious. We set $\epsilon = 0.1$ in the following experiments.

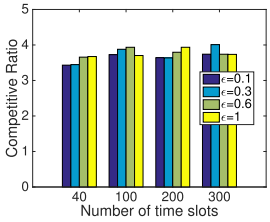

Fig. 5. Performance of ORFA with different # of time slots.

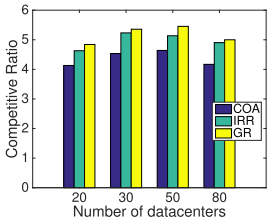

Fig. 6. Performance of COA with different # of datacenters.

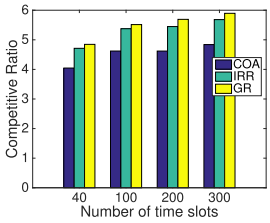

Fig. 7. Performance of COA with different # of time slots.

C. Performance of COA

We next evaluate ratios achieved by our complete online algorithm, COA in Alg. 1, derived by dividing the overall cost achieved by the integer solution of COA by the offline minimum overall cost of (7). We compare the ratios achieved by our algorithm and those of two other algorithms: (i) *IRR*, our online algorithm ORFA in Alg. 1 together with a randomized *independent* rounding algorithm, which simply rounds the VNF quantities to nearest integers (*e.g.*, 1.4 to 1), and computes the ratio only if the result flow routing is feasible; and (ii) *GR*, ORFA with a greedy rounding algorithm which rounds up the fractional VNF quantities to guarantee feasibility. Fig. 6 and Fig. 7 show that our online algorithm with the dependent rounding approach consistently outperforms the other algorithms. To recap, the ORFA provides us with a promising online fractional solution with good competitive ratios regardless of the number of time slots and datacenters (Fig. 4 and Fig. 5). Furthermore, the dependent rounding algorithm outperforms the other rounding algorithms (Fig. 6 and Fig. 7), such that the overall competitive ratio with the feasible integral solution is also promising.

D. Performance Under Different Shock Levels

Fig 8 plots the ratios under different shock levels of the flash crowds. As expected, a higher shock level brings larger ratios. However, even when the shock level increases significantly from 1 to 100, our ratios only increase from about 1 to around 6, which shows the stability of the performance of our algorithms even in extreme scenarios. Note that the COA performs worse than the ORFA, as the ORFA uses the fractional optimal solutions obtained by relaxing our original integer programming problem.

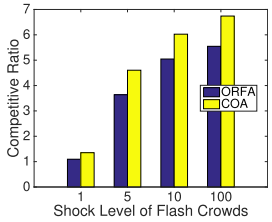

Fig. 8. Performance of our algorithms at various shock levels

VII. Concluding Remarks

The fast adoption and development of Network Function Virtualization solutions introduce a series of challenges in modern datacenter management. This work aims to efficiently manage cloud resources for VNF deployment to realize the NFV goals of significant cost reduction and ease of management, and also guarantee short end-to-end delay experienced by the flows. We first leverage a regularization method from online learning to reshape the relaxation of the original offline optimization problem. Then the new problem can be decomposed into a set of sub-problems, each of which can be solved optimally in polynomial time. Furthermore, we design an online dependent rounding scheme to obtain the final randomized mixed integer solution. We show that the final solution to the original problem yields a competitive ratio which is independent of the number of flows or time horizon, based on the analysis via a primal-dual framework. The cost effectiveness is further validated by both theoretical proof and a series of trace-driven simulations.

References

[1] "Network functions virtualization," Netw. Funct. Ind. Specification Group, White Paper, Oct. 2013. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] E. P. Quinn and E. T. Nadeau, *Problem Statement for Service Function Chaining*, document IETF RFC-Informational, 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc7498/

[3] A. Gember-Jacobson *et al.*, "OpenNF: Enabling innovation in network function control," in *Proc. ACM SIGCOMM*, 2014, pp. 163–174.

[4] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. (Nov. 2015). "On orchestrating virtual network functions in NFV." [Online]. Available: https://arxiv.org/pdf/1503.06377.pdf

[5] *IP Multimedia Subsystem*. [Online]. Available: https://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[6] N. Buchbinder, S. Chen, and J. S. Naor, "Competitive analysis via regularization," in *Proc. ACM SODA*, 2014, pp. 436–444.

[7] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *J. ACM*, vol. 53, pp. 324–360, May 2006.

[8] V. S. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "A unified approach to scheduling on unrelated parallel machines," *J. ACM*, vol. 56, no. 5, p. 28, Aug. 2009.

[9] S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma, *Service Function Chaining Use Cases In Data Centers*, document IETF draft-ietf-sfc-dc-use-cases-06, 2015. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sfc-dc-use-cases/

[10] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. of 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 459–473.

[11] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: High performance and flexible networking using virtualization on commodity platforms," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 1–15.

[12] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, p. 24.
[13] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 227–240.
[14] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.
[15] A. Gember *et al.* (2014). "Stratos: A network-aware orchestration layer for middleboxes in the cloud." [Online]. Available: https://arxiv.org/abs/1305.0209
[16] S. Palkar *et al.*, "E2: A framework for NFV applications," in *Proc. 25th ACM Symp. Operat. Syst. Principles (SOSP)*, 2015, pp. 121–136.
[17] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM)*, 2014, pp. 1–6.
[18] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. (Mar. 2015). "On Orchestrating Virtual Network Functions in NFV." [Online]. Available: https://arxiv.org/abs/1503.06377
[19] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.
[20] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 1346–1354.
[21] B. Zhang, J. Hwang, and T. Wood, "Toward online virtual network function placement in software defined networks," in *Proc. IEEE/ACM IWQoS*, Jun. 2016, pp. 1–6.
[22] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE ICNP*, Nov. 2016, pp. 1–10.
[23] M. Ghaznavi *et al.*, "Elastic virtual network function placement," in *Proc. IEEE CloudNet*, Oct. 2015, pp. 255–260.
[24] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.
[25] X. Wang *et al.*, "Online VNF scaling in datacenters," in *Proc. IEEE Cloud*, Jun./Jul. 2016, pp. 140–147.
[26] *Provider Provisioned Virtual Private Network (VPN) Terminology*. Accessed: Mar. 2005. [Online]. Available: https://www.ietf.org/rfc/rfc4026.txt
[27] *Benchmarking Terminology for Firewall Performance*. Accessed: Aug. 1999. [Online]. Available: https://tools.ietf.org/html/rfc2647
[28] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1098–1106.
[29] R. Cziva, S. Jouet, and D. P. Pezaros, "Roaming edge vNFs using glasgow network functions," in *Proc. ACM SIGCOMM*, 2016, pp. 601–602.
[30] *Amazon EC2 Pricing*. Accessed: 2018. [Online]. Available: http://aws.amazon.com/ec2/pricing/
[31] B. Zhang *et al.*, "Measurement-based analysis, modeling, and synthesis of the Internet delay space," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 229–242, Feb. 2010.
[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[33] T. Carnes and D. Shmoys, "Primal-dual schema for capacitated covering problems," in *Proc. Int. Conf. Integer Program. Combinat. Optim.*, 2008, pp. 288–302.
[34] N. Buchbinder, S. Chen, J. Naor, and O. Shamir, "Unified algorithms for online learning and competitive analysis," in *Proc. Workshop Conf.*, 2012, pp. 5-1–5-15.
[35] *AWS Global Infrastructure—Amazon Web Services*. Accessed: 2018. [Online]. Available: http://aws.amazon.com/about-aws/global-infrastructure/
[36] *Data Center Locations—Data Centers—Google*. Accessed: 2018. [Online]. Available: http://www.google.com/corporate/datacenter/locations.html
[37] *The Cogent's Network Map*. Accessed: 2009. [Online]. Available: http://cogentco.com/en/network/network-map
[38] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Comput. Netw.*, vol. 54, no. 5, pp. 877–878, 2010.
[39] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "Modeling, analysis and simulation of flash crowds on the Internet," in *Proc. IEEE MASCOTS*, Feb. 2003, pp. 1–15.
[40] *Global Internet Report 2015*. Accessed: 2017. [Online]. Available: http://www.internetsociety.org/globalinternetreport/

Yongzheng Jia received the B.E. degree in computer science from Yao Class, Tsinghua University, China, where he is currently pursuing the Ph.D. degree with the Institute of Interdisciplinary Information Sciences. His current research interests include cloud computing, online dating, and online education. He focuses on solving practical problems by using data science, combinatorial optimization, and game theory.

Chuan Wu (SM'16) received the B.Engr. and M.Engr. degrees from the Department of Computer Science and Technology, Tsinghua University, China, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2008. Since 2008, she has been with the Department of Computer Science, The University of Hong Kong, where she is currently an Associate Professor. Her current research is in the areas of cloud computing, distributed machine learning/big data analytics systems, and network function virtualization. She is a member of the ACM. She was a co-recipient of the Best Paper Awards of HotPOST 2012 and ACM e-Energy 2016.

Zongpeng Li received the B.E. degree from Tsinghua University in 1999, and the M.S. and Ph.D. degrees from the University of Toronto, in 2001 and 2005, respectively. His research interests are in computer networks, network coding, cloud computing, and energy networks. He was named an Edward S. Rogers Sr. Scholar in 2004, received the Alberta Ingenuity New Faculty Award in 2007, and was nominated for the Alfred P. Sloan Research Fellow in 2007. He coauthored papers that received Best Paper Awards at PAM 2008, HotPOST 2012, and ACM e-Energy 2016. He received the Outstanding Young Computer Science Researcher Prize from the Canadian Association of Computer Science in 2015.

Franck Le received the Ph.D. degree from Carnegie Mellon University and the Diplôme d'Ingénieur from the Ecole Nationale Superieure des Telecommunications de Bretagne in France. He is currently a Researcher with the IBM T. J. Watson Research Center.

Alex Liu received the Ph.D. degree in computer science from The University of Texas at Austin in 2006. His research interests focus on networking and security. He is currently an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and an Area Editor of *Computer Communications*. He received the IEEE & IFIP William C. Carter Award in 2004, a National Science Foundation CAREER Award in 2009, and the Michigan State University Withrow Distinguished Scholar Award in 2011. He also received Best Paper Awards from ICNP-2012, SRDS-2012, and LISA-2010.