Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds

Song Yang, *Member, IEEE*, Fan Li, *Member, IEEE*, Stojan Trajanovski, *Member, IEEE*, Xu Chen, *Member, IEEE*, Yu Wang, *Fellow, IEEE*, and Xiaoming Fu, *Senior Member, IEEE*

Abstract—Mobile Edge Computing (MEC) offers a way to shorten the cloud servicing delay by building the small-scale cloud infrastructures at the network edge, which are in close proximity to the end users. Moreover, Network Function Virtualization (NFV) has been an emerging technology that transforms from traditional dedicated hardware implementations to software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application. NFV therefore allows to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can largely reduce the overall costs in MEC. In this paper, we study the problem of how to place VNFs on edge and public clouds and route the traffic among adjacent VNF pairs, such that the maximum link load ratio is minimized and each user's requested delay is satisfied. We consider this problem for both totally ordered SFCs and partially ordered SFCs. We prove that this problem is NP-hard, even for the special case when only one VNF is requested. We subsequently propose an efficient randomized rounding approximation algorithm to solve this problem. Extensive simulation results show that the proposed approximation algorithm can achieve close-to-optimal performance in terms of acceptance ratio and maximum link load ratio.

Index Terms—Network function virtualization, mobile edge computing, delay, placement, routing

1 Introduction

Cloud computing [1] is a distributed computing and storage paradigm, which can provide virtually unlimited scalable service over the Internet for on-demand data-intensive applications. Data centers are usually the cloud computing-enabled infrastructures, and are located on some core router nodes in backbone networks. However, delay-sensitive applications for remote end users suffer from the long-distance network transmission delay, which remains a crucial drawback for cloud computing. The concept of Mobile Edge Computing (MEC) [2] has been proposed to bring the computing resources closer to end users by installing small resource-limited cloud infrastructures at the network edge (also called edge clouds), so as to provide delay-guaranteed services to end users.

Moreover, in the traditional network services provisioning paradigm, network functions (e.g., firewall or load balancer) which are also called middleboxes are usually implemented by the dedicated hardware appliances. Deploying hardware middleboxes is costly due to their high cost for design and production and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Network Function Virtualization (NFV) which is first proposed by the European Telecommunications Standards Institute (ETSI) [3], [4] has been emerged as an appealing solution, since it enables to replace dedicated hardware implementations with software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is therefore defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application [5]. NFV allows to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can largely reduce the overall costs.

Applying NFV to MEC will not only shorten servicing delay for end users but service providers will also benefit from lower expenditures and higher efficiency. The user requested service, in this context, consists of a sequence of VNFs that need to be placed on edge or public clouds. Considering that the edge cloud has limited capacity, it is also suggested [6], [7], [8], [9] to connect the edge clouds together in order to expose services to more nearby end users by

- *S. Yang and F. Li are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: {S.Yang, fli}@bit.edu.cn.*
- *S. Trajanovski is with Philips Research, Eindhoven 5656 AE, The Netherlands. E-mail: stojan.trajanovski@philips.com.*
- *X. Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China. E-mail: chenxu35@mail.sysu.edu.cn.*
- *Y. Wang is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA. E-mail: wangyu@temple.edu.*
- *X. Fu is with the Institute of Computer Science, University of Göttingen, Göttingen 37077, Germany. E-mail: Fu@cs.uni-goettingen.de.*

Manuscript received 15 May 2019; revised 23 Aug. 2019; accepted 12 Sept. 2019. Date of publication 19 Sept. 2019; date of current version 7 Jan. 2021. (Corresponding author: Song Yang.)
Digital Object Identifier no. 10.1109/TMC.2019.2942306

1536-1233 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

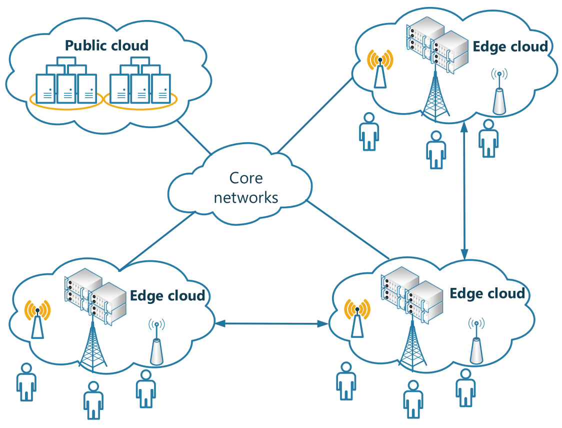


Fig. 1. A mobile edge computing framework.

efficiently utilizing and sharing the capacity and load of edge clouds. Even this, the connected edge clouds may sometimes fail to satisfy all the end users when users' requested resources increase. Hence, the edge clouds also need to collaborate with remote public cloud with sufficient capacity by placing requested VNFs on it for the users who ask for delay-tolerant services via e.g., long distance core networks, which can be shown in Fig. 1. This deals with where to place the VNFs and how to route the traffic by selecting appropriate paths during the whole SFC according to the user's delay requirement. In this paper, we study how to place each user's requested VNFs on edge and public clouds, and route the packets among these VNFs in order to minimize the maximum link load ratio and meet each user's delay requirement. Our key contributions are as follows:

- We analyze the traversing delay calculation in edge clouds for both totally ordered SFC and partially ordered SFC.
- We define the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds and prove it is NP-hard.
- We propose a randomized rounding approximation algorithm to solve the DVPR problem.
- We conduct extensive simulations to validate the performances of the proposed algorithm with three heuristics.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 models traversing delay for both totally ordered SFC and partially ordered SFC in edge clouds. Section 4 defines the delay-aware VNF placement and routing problem in edge clouds, formulates it as an exact INLP and proves it is NP-hard. In Section 5, we propose an approximation algorithm to solve this problem. Section 6 provides the simulation results and we conclude in Section 7.

2 Related Work

A survey about computation offloading and computation resources allocation in MEC can be found in [10]. Mao et al. [11] provide an overview about the communication models and resource management in MEC such as MEC server scheduling and cooperative computing. Moreover, a comprehensive survey about NFV can be found in [12], [13], [14]. Herrera and Botero [15] provide a survey about resource allocation in NFV.

2.1 Traffic/Cost-Aware VNF Placement and Routing in Generic Networks

Mehraghdam et al. [16] first explores the VNF placement problem by presenting an ILP to solve this problem. Eramo et al. [17] target the VNF placement and routing problem with the goal of maximizing the amount of data that can be processed within the network at peak traffic time interval. They subsequently study how to consolidate VNFs and shut down unused servers when traffic demands decrease such that the total operation cost (energy consumption+revenue loss) is minimized. They propose an exact Integer Linear Programming (ILP) and an efficient heuristic to solve the problem. Cohen et al. [18] propose a near-optimal approximation algorithm to place VNFs without considering network function dependency. Kuo et al. [19] relax/approximate the VNF placement and routing problem based on the intuition that placing VNFs on a shorter path consumes less link bandwidth, but might also reduce VM reuse opportunities; reusing more VMs might lead to a longer path, and so it consumes more link bandwidth. Guo et al. [20] jointly consider the VNF placement and routing problem in data centers. They propose a randomized approximation algorithm when the traffic matrix is known in advance and a competitive online algorithm when the future arriving traffic is not known. However, they assume that one configuration in data centers consists of one VNF placement and one routing path solution, and a (limited) set of configurations is given. The delay is not taken into account in these papers. Gupta et al. [21] develop a column-generation-based approaches to solve the VNF placement and routing problem. Liu et al. [22] present a column-generation-based approache to solve the SFC readjustment problem. Bhamare et al. [23] deal with VNF placement in a multi-cloud scenario with constraints of deployment costs and servicing delay by proposing an ILP and an affinity-based approach heuristic. A Minimum-Residue heuristic is also presented in [24] for VNF placement in multi-cloud scenario. However, the path selection problem among VNF pairs are not considered in [23], [24].

It is worthwhile to mention that the Virtual Machine (VM) placement problem is similar to the VNF placement. However, the communication/routing path between each VM pair do not necessarilty form a SFC, which makes it different from the VNF placement problem. Please see these works in e.g., [25], [26], [27], [28].

2.2 Delay-Aware VNF Placement and Routing in Generic Networks

Qu et al. [29] consider the VNF transmission and processing delays, and formulate the joint problem of VNF placement and routing as a Mixed Integer Linear Program (MILP). However, they assume that the virtual link between two physical nodes can at most process one traffic flow at one time. Alameddine et al. [30] jointly consider the network function mapping, traffic routing and the deadline-aware service scheduling problem, where it is assumed that the requested VNFs have already been placed on network nodes. They propose a tabu search-based algorithm and two straightforward heuristics to solve this problem. Zhang et al. [31] devise an Alternating Direction Method of Multipliers (ADMM)-based algorithm to solve the delay-aware VNF placement and routing problem without considering

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

the nodes' processing delay in their problem. Li et al. [32] address the delay-aware middlebox placement and routing problem by leveraging a packet queuing model. Sun et al. [33] implement a framework that enables (independent) network functions to work in parallel, which improves NFV performance in terms of delay, but this comes at the expense of increasing network resource (e.g., network bandwidth). By duplicating an original graph with m connected copy graphs, Huin et al. [34] present a mathematical formulation with the aid of column generation (using a limited number of configurations) that can scale well with problem inputs (e.g., number of requests or nodes). Allybokus et al. [35] propose an exact ILP and a greedy heuristic to solve the VNF placement and routing problem for both fully and partially ordered SFCs. However, their solutions only consider a simple path within a SFC. Dwaraki and Wolf [36] devise a layered-graph based heuristic to solve the delay-aware traffic routing problem when the VNFs are assumed to be placed on network nodes. Similarly, Pei et al. [37] devise a layered-graph based heuristic to jointly solve the delay-aware VNF placement and routing problem. In [38], we present an exact INLP and a recursive heuristic to solve the delay-aware and availability-aware VNF placement and routing problem in generic networks.

2.3 Delay-Aware VNF/Rule Placement and Routing in Edge Clouds and Software Defined Networks (SDN)

Hou et al. [39] suppose that the edge servers can host limited κ number of services. When the requested service does not exist in the edge servers, downloading this service from cloud incurs a cost of ω, otherwise the provisioning cost is negligible. Without any prior knowledge of future traffic, they propose an online algorithm to configure κ services on edge servers. Ma et al. [40] model the edge cloud network as a queuing network, and they formulate the system servicing delay problem as a convex optimization problem. They subsequently present an algorithm with the linear complexity to solve the proposed convex optimization problem. Moreover, Saha et al. [41] target the QoS-aware (such as delay and packet-loss) routing problem in software defined IoT networks by considering the SDN rule capacity constraint. They present an exact ILP as well as a Yen's k-shortest path-based heuristic. Bera et al. [42] present an adaptive flow-rule placement scheme which consists of three phases, namely (1) forwarding path selection, (2) flow-rule placement and (3) rule redistribution. To solve them, they respectively formulate (1) via a max-flow-min-cost problem, devise two greedy heuristic approaches for (2), and propose a rule redistribution scheme to detect rule congestion at a switch.

Cziva et al. [43] tackle the VNF placement problem at the network edge in order to minimize the total expected latency from all users to their respective VNFs. They further employ the Optimal Stopping Theory to determine when to re-evaluate the optimal placement problem by taking into account the migration cost and random path delay. However, they assume that only one VNF is enough to provide a service to one user instead of a sequence of VNFs. Yang et al. [9] study how to place NFV-enabled service on a minimum number of edge nodes and find routes from the access point to the requested service located node in order to meet the delay requirement. They propose a heuristic-based incremental allocation mechanism to solve this problem. Different form the work in [9], [43], we propose a general model to quantitatively calculate flow traversing delay of a SFC in edge clouds, and devise a randomized rounding algorithm to jointly solve the delay-aware VNF placement and traffic routing problem in edge clouds. Moreover, there are also some studies about VNF placement in virtual evolved packet core (4G/5G) networks [44], [45], Cloud-Radio Access Networks [46], etc.

In the previous works, either an exact ILP algorithm or an efficient heuristic is proposed to solve the general VNF placement and routing problem. However, the exact ILP solution has exponential running time which means that it cannot scale well especially when the problem size increases. The heuristics are shown to perform well in the simulations under certain parameter settings, but it has no performance guarantee, meaning that it may not always output a feasible solution even if the optimal solution exists. Our contribution is devising a randomized rounding approximation algorithm to (jointly) solve the delay-aware VNF placement and routing problem in edge clouds. Moreover, our work can also be adjusted to solve the delay-aware VNF placement and routing problem in generic networks.

3 Network Delay Model

3.1 Service Function Chaining

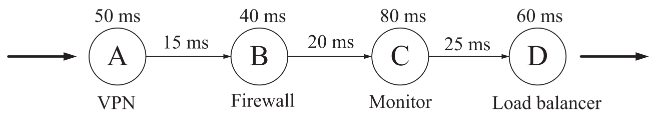
We consider traversing delays under two cases for VNFs in a SFC, namely, (1) Totally Ordered SFC: there is a total dependency order on the VNF set, and (2) Partially Ordered SFC: there exists dependency among a subset of VNFs. That is, there exists at least two VNFs, such that they have the same predecessor function and successor function but they have no dependencies with each other. The traversing delay in the totally ordered SFC is calculated as follows:

$$\sum_{f\in F, n\in\mathcal{N}} \Psi_n^f + \sum_{l\in p_k} T(l), \qquad (1)$$

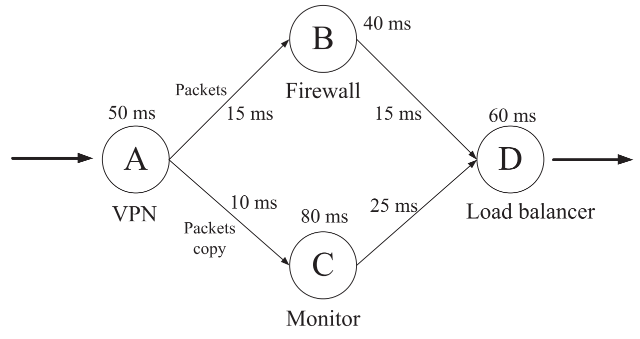
where Ψ_n^f represents the processing delay for function f on node n, p_k denotes the path that traverses each placed VNF, and $T(l)$ indicates the flow delivering delay on link l. For example, Fig. 2a is a totally ordered SFC and the traversing delay in this chain is $\underbrace{(50+40+80+60)}_{\text{total node delay}} + \underbrace{(15+20+25)}_{\text{total link delay}} =$ 290 ms.

However, since function monitor only maintains the packets and does not change the packets, firewall and monitor can work in parallel as shown in Fig. 2b. After that, firewall and monitor functions send their individual processed packets to load balancer. As a result, load balancer only needs to select the packets from firewall and process them. The example in Fig. 2b belongs to the partially ordered SFC. We partition the partially ordered SFC into several segments and thus there is one or more independent VNFs in each segment. For example, there are 3 segments in Fig. 2b: Segment 1 contains the function VPN, segment 2 is composed of functions firewall and monitor, and segment 3 has the function load balancer. As a result, the traversing delay in a partially ordered SFC is equal to the longest path delay

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.



(a) Totally Ordered SFC.



(b) Partially Ordered SFC.

Fig. 2. Totally ordered SFC versus partially ordered SFC, derived from [33].

from the node which hosts one of the VNFs in the first segment to the node which contains one of the VNFs in the last segment. For example, the traversing delay of SFC in Fig. 2b is equal to $\underbrace{(50+80+60)}_{\text{total node delay}} + \underbrace{(10+25)}_{\text{total link delay}} = 225$ ms, which is less than the one in Fig. 2a. This example also illustrates that the partially ordered SFC can reduce delay compared to the totally ordered SFC. We notice that the totally ordered SFC is a special case of the partially ordered SFC, since when each segment contains only one VNF, the partially ordered SFC becomes the totally ordered SFC. Without loss of generality, we assume a (totally ordered or partially ordered) SFC consists of a set of $|M|$ segments M, and each segment $m \in M$ has $|m|$ functions. We use m_i and m_j to represent two consecutive segments, where $j = i+1$ and $1 \leq i \leq |M|-1$. In order to calculate the traversing delay in a partially ordered SFC with segments M, we first make the definition of *totally ordered sub-SFC* as follows:

Definition 1 Totally Ordered Sub-SFC. *Suppose there is a partially ordered SFC $\mathcal{C}$ with segment set M. A totally ordered sub-SFC is a chaining consisting of exactly one VNF in each segment.*

According the above definition, there are in total $\prod_{i=1}^{|M|} |m_i|$ totally ordered sub-SFCs. For example in Fig. 2b, there are $1 \cdot 2 \cdot 1 = 2$ totally ordered sub-SFCs, namely $A \to B \to D$ and $A \to C \to D$. As a result, the delay value of a partially ordered SFC is the maximum delay value among all the totally ordered sub-SFCs composing it.

3.2 Traversing Delay in a SFC in Edge Clouds

Assume that there is a set of edge clouds denoted by $\mathcal{E}$ and one remote public cloud[1] represented by $\mathcal{C}$. Each edge cloud $E \in \mathcal{E}$ builds a mount of edge servers along with the base station in an area and serve the end users within the coverage of the base station. The users within the coverage of E is denoted by R_E. In this sense, requests sent from R_E should be first received by the edge servers in E, and if E cannot process these requests because of capacity limit, it can relay these requests to other edge clouds or public cloud. In this paper, we assume that any end user can only be within the coverage of one edge cloud. For simplicity, we assume that one end user sends only one request to the edge cloud. The notations used in this paper are summarized in Table 1.

Suppose there is a user u_r who is within the coverage of edge cloud E, and for simplicity u_r requests a fully ordered SFC consisting of h virtual network functions $f_1, f_2, \ldots, f_h$, where f_i is placed on node n_{f_i} for $1 \leq i \leq h$. If $T(n_i, n_j)$ denotes the traversing delay between edge or cloud nodes n_i and n_j, and Ψ_n^f indicates the VNF f's processing time on node n, then the total delay for serving u_r is[2]

$$T(E, n_{f_1}) + \sum_{i=1}^{h} \Psi_{n_{f_i}}^{f_i} + \sum_{j=1}^{h-1} T(n_{f_j}, n_{f_{j+1}}), \tag{2}$$

where $T(E, n_{f_1})$ indicates the traversing delay from E to n_{f_1}. $T(E, n_{f_1})$ means that if f_1 is not located in proximity to user in E, then we need to take into account the traversing delay from E to its located edge cloud or public cloud. $\sum_{i=1}^{h} \Psi_{n_{f_i}}^{f_i}$ calculates the total node processing time, and $\sum_{j=1}^{h-1} T(n_{f_j}, n_{f_{j+1}})$ calculates the total path delay. For example in Fig. 3, there are three edge clouds and one public cloud. Each edge cloud can only host one VNF because of limited capacity for simplicity, and the public cloud host two remaining VNFs. It is assumed that user 1 and user 2 belong to edge cloud E_1. Moreover, edge cloud E_2 and edge cloud E_3 serve user 3 and user 4, respectively. The requested VNFs (SFC) are also shown in Fig. 3. When calculating the traversing delay of requested SFC from the request of user 1, the first requested VNF is not located in edge cloud E_1, therefore we must calculate the delay from E_1 to E_2 (which is 12, since f_1 is placed on E_2), and then calculate the delay of SFC $f_1 - f_2 - f_3$, which is $\underbrace{(25+20+18)}_{\text{total node delay}} + \underbrace{(12+13)}_{\text{total link delay}} = 88$. As a result, the total delay for serving the request from user 1 is $12 + 88 = 100$. Analogously, the delay for serving user 4 is 155. On the other hand, the first requested VNF of user 2 and 3 is placed in the edge cloud they belong to, therefore serving user 2 consumes a delay of $20 + 15 + 80 = 115$, and serving user 3 incurs a delay of $25 + 20 + 12 = 57$.

4 Problem Definition and Complexity Analysis

4.1 Problem Definition

There is a set of edge clouds $\mathcal{E}$, a set of switch/router nodes $\mathcal{S}$ and a remote cloud $\mathcal{C}$. Each edge cloud $n \in \mathcal{E}$ consisting of a limited number of edge servers has a total capacity of $\pi(n)$, and the public cloud $\mathcal{C}$ is assumed to have infinite capacity.

1. For simplicity, we only assume one public cloud in this paper, but our work can also be extended to the case of multiple public clouds.

2. When an end user requests a partially ordered SFC, we can calculate the delay of each totally ordered sub-SFC according to Eq. (2) and take the maximum value as the final delay value.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Notations

Notation	Description
$\mathcal{E}, \mathcal{S}, \mathcal{C}$,	The set of edge clouds, switch nodes and public cloud
$\mathcal{N}$	The set of N nodes that can host VNFs, i.e, $\mathcal{N} = \mathcal{E} \cup \mathcal{C}$
$\mathcal{L}, b(l), t(l)$	The set of L links, traffic load and delay of link $l \in \mathcal{L}$
$\pi(n), c(l)$	Capacity of node $n \in \mathcal{N}$ and link $l \in \mathcal{L}$
$\mathcal{G}(\mathcal{N}, \mathcal{L})$	A network with set of nodes and links $\mathcal{N}$ and $\mathcal{L}$
$P^{u,v}, K$	Path set between u and v, the number of paths in the set
$T(p_k)$	Delay of path p_k
R	The set of requests. For each $r(E, \delta, F, D) \in R$, E indicates the edge cloud the request belongs to, δ represents the data transmitting rate, F denotes a set of requested VNFs with predefined order, D means the requested delay
m_i, m_j	Two consecutive segments for r, where $j = i + 1$
f_i, f_j	The ith and jth requested VNF for a totally ordered (sub-)SFC, where $j = i + 1$
$\mathbb{B}_r$	The set of totally ordered sub-SFCs of request r
$\mathbb{F}$	A set of total requested VNFs
$\eta(f)$	The required processing capacity of VNF f
Ψ_n^f	Processing time for VNF f on node n
$H_{l,k}^{u,v}$	A given boolean array indicating whether link l is traversed by path p_k between u and v
λ	A fractional variable meaning maximum link load ratio
$X_n^{r,f}$	A boolean variable. It is 1 if r's requested VNF f is placed on n; and 0 otherwise
$Y_{u,v,k}^{r,i,j}$	A boolean variable. It is 1 if r's requested f_i and f_j are placed on node u and v, respectively, and $p_k \in P^{u,v}$ is selected; and 0 otherwise.

Switch nodes are used to forward the traffic. Assume there is a set $\mathcal{L}$ of L links to connect different nodes in $\mathcal{E} \cup \mathcal{S} \cup \mathcal{C}$. For ease of presentation, we let $\mathcal{N} = \mathcal{E} \cup \mathcal{C}$ denote the node set that can host VNFs, and we use $\mathcal{N}$ (which does not contain the switch node set) without loss of generality to stand for the node set in the network. Each VNF $f \in \mathbb{F}$ on node $n \in \mathcal{N}$ requires time of Ψ_n^f to process it, and we use $\eta(f)$ to denote its processing capacity. Each link $l \in \mathcal{L}$ has a capacity of $c(l)$, and traversing delay of $t(l)$. Let $b(l)$ denote the total traffic load on l, then the maximum link load ratio λ is defined as: $\max\{\frac{b(l)}{c(l)}, l \in \mathcal{L}\}$ and it is a single variable. For each node pair (u, v) where $u, v \in \mathcal{N}$, we assume that the path set between them is known[3] and denoted by $P^{u,v}$ with a number of K paths. R represents a set of $|R|$ requests, and for each request $r(E, \delta, F, D) \in R$, E denotes the edge cloud the request belongs to, δ stands for the data transmitted rate, $F \subset \mathbb{F}$ indicates a set of $|F|$ requested VNFs with predefined order (totally or partially ordered), and D stands for the total requested end-to-end delay. Similar to [9], [30], [31], [34], [37], we assume that R is known or given. Formally, the Delay-aware VNF Placement and Routing problem in edge clouds can be defined as follows:

Definition 2. *Given are a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$, and a set of requests R. For each request $r(E, \delta, F, D) \in R$, the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds is to place its requested VNFs on $\mathcal{N}$ and find routes among each adjacent VNF pair such that λ is minimized and the total flow traversing delay is no greater than D.*

The purpose of minimizing λ is to avoid the network bottleneck, e.g., one or more links are highly loaded and while some other links are less loaded. In network bottleneck scenario, the highly loaded link(s) are unavailable to transport more data from a source to a destination because of the lack of free capacity. In the worst case, there may not exist bandwidth-free path from a specified source to a specified destination if these (critical) highly loaded links are not available, or it may take longer delay by traversing some more links from a source to a destination. Nevertheless, we mention that our problem definition/formulation and proposed approximation algorithm are general and flexible, since we could also remove the objective or change to select some other objectives. More specifically, if we remove the objective, then the problem becomes to find the VNF placement and traffic routing for each request in edge clouds without violating the specified delay requirement. And the respective approximation algorithm proposed in Section 5 can be slightly adjusted (by removing the objective and λ-related constraints) to solve it as well.

4.2 An Exact Formulation

In this subsection, we formulate the DVPR problem as an exact Integer Nonlinear Programming (INLP) formulation. We begin with some necessary notations and variables.

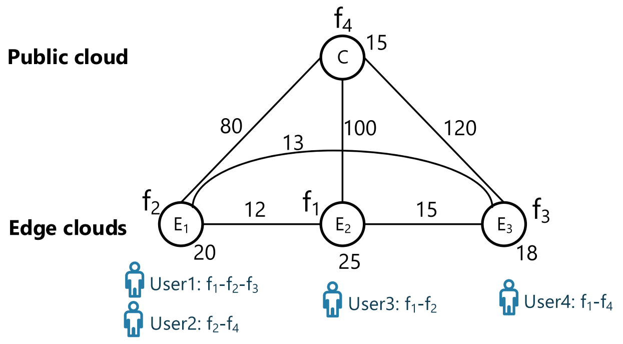


Fig. 3. An example of calculating the delay in a SFC in edge clouds.

3. According to [47], at most 6 paths in GÉANT network are enough for serving 11460 traffic matrices during the entire 4 month duration without violating Quality of Service (QoS). We therefore assume that a (small) set of paths is sufficient for calculating the optimal solution. This set of paths is precalculated and given in the problem.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

INLP notations:

$H^{u,v}_{l,k}$: A given boolean array. It is 1 (true) if link l is traversed by the path p_k between u and v, and 0 (false) otherwise.

$T(p_k)$: Delay of path p_k.

$\mathbb{F}$: The set of total requested $|\mathbb{F}|$ VNFs. For ease of calculation of traversing delay, for each request r, we add two dummy VNFs f_0 as the first VNF of the chain and $f_{|\mathbb{F}|+1}$ as the last VNF of the chain, with $\eta(f_0) = \eta(f_{|\mathbb{F}|+1}) = 0$ and $\Psi^{f_0}_n = \Psi^{f_{|\mathbb{F}|+1}}_n = 0$, where $n \in \mathcal{N}$.

$\mathbb{B}_r$: The set of totally ordered sub-SFCs of request r.

f_i, f_j: The ith and jth requested VNF for r, where $0 \le i \le |\mathbb{F}|, j = i+1$ (two consecutive VNFs).

INLP variables:

λ: A float variable ranging in [0,1] that represents the link load factor or ratio.

$X^{r,f}_n$: A boolean variable. It is 1 (true) if r's requested VNF f is placed on n; and 0 (false) otherwise.

$Y^{r,i,j}_{u,v,k}$: A boolean variable. It is 1 (true) if r's requested f_i and f_j are placed on node u and v, respectively, and $p_k \in P^{u,v}$ is selected; and 0 (false) otherwise.

Objective:

$$\min \quad \lambda \tag{3}$$

Placement Constraints:

$$\sum_{n\in\mathcal{N}} X^{r,f}_n = 1 \quad \forall r \in R : f \in r.F \tag{4}$$

$$\sum_{u,v} \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} = 1 \quad \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B. \tag{5}$$

Path Selection Constraint:

$$X^{r,f_i}_u \cdot X^{r,f_j}_v = \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} \quad \forall r \in R, B \in \mathbb{B}_r, \\ f_i, f_j \in B, u, v \in \mathcal{N} \tag{6}$$

Delay Constraint:

$$\sum_{f_i,f_j\in B} \sum_{u,v\in\mathcal{N}} \left(X^{r,f_i}_u \cdot X^{r,f_j}_v \cdot \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} \cdot \left(T(p_k) + \frac{\Phi^{f_i}_u + \Phi^{f_j}_v}{2} \right)\right) \\ \le D \quad \forall r \in R, B \in \mathbb{B}_r. \tag{7}$$

Edge Cloud Capacity Constraint:

$$\sum_{f\in\mathbb{F}} \max\nolimits_{r\in R: f\in r.F} X^{r,f}_n \cdot \eta(f) \le \pi(n) \quad \forall n \in \mathcal{N}\backslash\mathcal{C}. \tag{8}$$

Link Capacity Constraint:

$$\sum_{r\in R} \sum_{B\in\mathbb{B}_r} \sum_{f_i,f_j\in B} \sum_{u,v\in\mathcal{N}} \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} \cdot H^{u,v}_{l,k} \cdot r.\delta \le \lambda \cdot c(l) \\ \forall l \in \mathcal{L} \tag{9}$$

Eq. (3) minimizes the maximum link load ratio. Eq. (4) ensures that for each user's requested VNF f, it must be placed on one node in the network. Eq. (5) ensures that for each requested VNF pair f_i, f_j in a totally ordered sub-SFC, they must be placed on one or two nodes. The above two placement constraints are set for variables $X^{r,f}_n$ and $Y^{r,i,j}_{u,v,k}$, respectively, and Eq. (6) establishes the equality relation between $X^{r,f}_n$ and $Y^{r,i,j}_{u,v,k}$. More specifically, Eq. (6) indicates that when request r places f_i and f_j on u and v, respectively, only one path between u and v can be selected to use. Eq. (7) ensures that for each request the total traversing delay in each sub-ordered SFC is no greater than D. In particular, $\sum_{f_i,f_j}(\Phi^{f_i}_u + \Phi^{f_j}_v)/2$ calculates the total node processing delay. This is because $f_1, f_2, \ldots$ have been counted twice in Eq. (7), we take the sum of $\Phi^{f_i}_u + \Phi^{f_j}_v$ and let it be divided by 2. Eq. (8) ensures that each edge cloud's capacity is not violated. Eq. (9) ensures that each link's load does not exceed to λ times its total capacity.

4.3 Complexity Analysis

Theorem 1. *The DVPR problem is NP-hard, even when* $|\mathbb{F}| = 1$ *and* $|R| \ge 2$.

Proof. Since totally ordered SFC is a special case of partially ordered SFC, we only analyze the problem complexity for the totally ordered SFC without loss of generality. Ma et al. [48] prove that the general VNF placement and routing problem can be reduced to the NP-hard Hamiltonian cycle problem [49], which means that the problem is also NP-hard. Next, we will prove that even when $|\mathbb{F}| = 1$ and $|R| \ge 2$, the DVPR problem is still NP-hard.

Assuming that there are two edge clouds, E_1 and E_2. Two user requests r_1 and r_2 which are both in proximity of E_1 ask for VNF f_1 and f_2, respectively. It is assumed that E_1 has no spare capacity and therefore f_1 and f_2 have to be placed on E_2. Moreover, we assume $c(l) = r_1.\delta = r_2.\delta, \forall l \in \mathcal{L}$. In this sense, each link can only be traversed by at most one time and $\lambda = 1$ in the optimal solution. Hence, the DVPR problem is to find two link-disjoint paths from E_1 to E_2 such that each path delay is no greater than the specified delay value. Now, the DVPR problem is equivalent to the NP-hard min-max problem [50], which is to find two link-disjoint paths from a source to a destination, such that the longer path delay (denoted by T_x) is minimized, if we assume $\max(r_1.D, r_2.D) = T_x$. The proof is therefore complete. □

5 Approximation Algorithm

Considering that the exact INLP solution has exponential running time [51] because of its boolean variables and nonlinear constraints, it cannot scale well especially when the problem size increases. In this section we propose a polynomial-time approximation algorithm to solve the DVPR problem. We first transform the exact INLP in Eqs. (3), (4), (5), (6), (7), (8), and (9) to a Linear Programming (LP). After solving the LP, we subsequently achieve a solution via the randomized rounding method with proved bounded approximation ratio.

5.1 Transformation from the INLP to the LP

First, using Eq. (6) into Eq. (7) we get

$$\sum_{f_i,f_j\in B} \sum_{u,v\in\mathcal{N}} \left(\sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} \cdot \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k} \cdot \left(T(p_k) + \frac{\Phi^{f_i}_u + \Phi^{f_j}_v}{2} \right)\right) \\ \le D \, \forall r \in R, B \in \mathbb{B}_r. \tag{10}$$

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

It is important to notice that $Y_{u,v,k}^{r,i,j}$ takes binary values in $\{0,1\}$ and if we decompose the two innermost sum into products we get pairwise products in $Y_{u,v,k}^{r,i,j}$. However, according to the placement constraint (Eq. (5)) there is a single $Y_{u,v,k}^{r,i,j}$ equals to 1 and all the rest are 0 for a fixed r. Hence, all the pairwise products of $Y_{u,v,k}^{r,i,j}$ in Eq. (10) are 0, except for one that is $(Y_{u,v,k}^{r,i,j})^2 = Y_{u,v,k}^{r,i,j}$. Finally, Eq. (10) boils down to the following equivalent constraint of Eq. (7):

$$\sum_{f_i,f_j \in B} \sum_{u,v \in \mathcal{N}} \left(\sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot \left(T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2} \right) \right) \leq D \tag{11}$$
$$\forall r \in R, B \in \mathbb{B}_r.$$

Second, considering that Eq. (6) is nonlinear, we transform it to the following equivalent linear constraints without losing any accuracy

$$\begin{cases} X_u^{r,f_i} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_v^{r,f_j} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_u^{r,f_i} + X_v^{r,f_j} - 1 \leq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B, u, v \in \mathcal{N} \end{cases} \tag{12}$$

After the transformation, we have the following relaxed Linear Programming (LP) to solve the DVPR problem, where $Y_{u,v,k}^{r,i,j}$ and $X_n^{r,f}$ are set to be a fractional number ($\in [0,1]$)

$$min \ \ \lambda$$
$$s.t. \begin{cases} \sum_{n \in \mathcal{N}} X_n^{r,f} = 1 \ \ \forall r \in R, f \in r.F \\ \sum_{u,v} \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} = 1 \ \ \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B \\ X_u^{r,f_i} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_v^{r,f_j} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_u^{r,f_i} + X_v^{r,f_j} - 1 \leq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B, u, v \in \mathcal{N} \\ \sum_{f_i,f_j \in B} \sum_{u,v \in \mathcal{N}} \left(\sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot \left(T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2} \right) \right) \\ \leq D \ \ \forall r \in R, B \in \mathbb{B}_r \\ \sum_{f \in \mathbb{F}} \max_{r \in R: f \in r.f} X_n^{r,f} \cdot \eta(f) \leq \pi(n) \ \ \forall n \in \mathcal{N} \backslash \mathcal{C} \\ \sum_{r \in R} \sum_{B \in \mathbb{B}_r} \sum_{f_i,f_j \in B} \sum_{u,v \in \mathcal{N}} \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot H_{l,k}^{u,v} \cdot r.\delta \\ \leq \lambda \cdot c(l) \ \ \forall l \in \mathcal{L} \end{cases} . \tag{13}$$

5.2 Randomized Rounding Approximation Algorithm

In this subsection, we present the Randomized Rounding VNF placement and routing Algorithm (RRVA) in Algorithm 1 to solve the DVPR problem.

The rational of RRVA in Algorithm 1 is that after solving LP in Eq. (13), we first get fractional solutions $\widetilde{Y_{u,v,k}^{r,i,j}}$ (and $\widetilde{X_n^{r,f}}$). Our aim is to derive an integer solution from $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ by randomized rounding. To that end, RRVA first uses $Ht[R][F]$ to store which node to host the requested VNF f of r, and uses $Rt[R][F][F]$ to store which path to route traffic from r's requested f_i to f_j. When f_i and f_j are placed on the same node, $Rt[r][f_i][f_j] = 0$. Moreover, $At[R]$ represents whether r is successfully served, and initially is set to be true for all the requests. After that, for each requested totally ordered sub-SFC $B \in B_r$ of request r, and for each its requested VNF pair $(f_i, f_j) \in B$, RRVA finds the appropriate nodes for hosting f_i and f_j and the path to route the traffic between them if they are placed on different nodes. Let n_x be the node where f_j is placed on in each iteration. In particular, when f_i and f_j have already been placed on the nodes in the previous iterations by other totally ordered sub-SFC, Step 8 assigns n_x with $Ht[r][f_j]$. Otherwise, it indicates that at least f_j is not placed on the network. When $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$, RRVA uses P to store the results where f_i and f_j are placed on the same node, and uses Q to store the results where f_i and f_j are placed on different nodes. If P is not empty, RRVA selects one $\widetilde{Y_{u,v,k}^{r,i,j}}$ from P where the node u has maximum residual capacity to host f_i and f_j. However, when $Ht[r][f_i] \neq null$, it indicates that f_i has already been placed on the network by other totally ordered sub-SFC determined by RRVA, and therefore we need to select $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ from P where $u = Ht[r][f_i]$. The reason is that one VNF can only be placed on one node. If P is empty and Q is not empty, RRVA randomly selects one $\widetilde{Y_{u,v,k}^{r,i,j}}$ from Q. Similarly, when $Ht[r][f_i] \neq null$ we need to select $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ from Q where $u = Ht[r][f_i]$. Because f_i except for f_0 and $f_{|\mathbb{F}|+1}$ occurs twice in the algorithm, we need to decrease the capacity of its hosted node only once, which can be seen in steps 18 and 21. Accordingly, RRVA prunes the link bandwidth that is traversed by the selected path p_k between u and v. The above procedure continues until all the requests have been iterated.

Algorithm 1. RRVA $(\mathcal{G}(\mathcal{N}, \mathcal{L}), R, \mathcal{F})$

```
1:  Solve the LP in Eq. (13) to obtain $\widetilde{Y_{u,v,k}^{r,i,j}}$
2:  $Ht[R][F] \leftarrow null$, $Rt[R][F][F] \leftarrow 0$, $At[R] \leftarrow true$
3:  foreach request $r \in R$ do
4:      foreach $B \in B_r$ do
5:          $n_x \leftarrow r.E$
6:          foreach $f_i, f_j \in B$ do
7:              if $Ht[r][f_i] \neq null$ && $Ht[r][f_j] \neq null$ then
8:                  $n_x \leftarrow Ht[r][f_j]$
9:              else
10:                 $P \leftarrow \emptyset, Q \leftarrow \emptyset$
11:                 if $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ && $u == n_x$ then
12:                     if $u == v$ then
13:                         $P.Add(\widetilde{Y_{u,v,k}^{r,i,j}})$
14:                     else
15:                         $Q.Add(\widetilde{Y_{u,v,k}^{r,i,j}})$
16:                 if $P \neq \emptyset$ then
17:                     Pick $v$ with maximum node capacity from $P$
                        such that $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ and $u == Ht[r][f_i]$ if
                        $Ht[r][f_i] \neq null$
18:                     $\pi(v) \leftarrow \pi(v) - \eta(f_j), Ht[r][f_i] \leftarrow u, Ht[r][f_j] \leftarrow v,$
                        $Rt[r][f_i][f_j] \leftarrow 0$
19:                 else if $Q \neq \emptyset$ then
20:                     Randomly select $\widetilde{Y_{u,v,k}^{r,i,j}} > 0$ from $Q$ such that
                        $u == Ht[r][f_i]$ if $Ht[r][f_i] \neq null$
21:                     $\pi(v) \leftarrow \pi(v) - \eta(f_j), n_x \leftarrow v,$   $Ht[r][f_i] \leftarrow u,$
                        $Ht[r][f_j] \leftarrow v, Rt[r][f_i][f_j] \leftarrow p_k$
22:                     Prune the link bandwidth according to the
                        selected path $p_k$ based on $\widetilde{Y_{u,v,k}^{r,i,j}}$
```

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

The time complexity of RRVA can be analyzed like this: The time complexity of RRVA is dominated by the LP in Eq. (13). There exists efficient polynomial time algorithm to solve the LP with the current best worst-case complexity of $O([\frac{I^3}{\ln I}]\gamma)$ by an interior-point method according to [52], where I is the number of variables and γ is the bit size of the problem (related to the number of bits in its binary representation). There are in total $O(|R||F|N + K|R||F|^2N^2) = O(K|R||F|^2N^2)$ variables in Eq. (13), leading to a total time complexity of $O(\frac{\gamma K^3|R|^3|F|^6N^6}{ln(K|R||F|^2N^2)})$ for RRVA, which indicates it has polynomial running time.

5.3 Approximation Performance Analysis

The analysis leverages on the method of Upper Tail Chernoff bound [53] and Union Bound (Boole's inequality) [54, Chapter 4.7] — a technique often used in other works (see e.g., [55], [56]). For completeness, we first give a formal definition of the Upper Tail Chernoff bound and Union Bound inequality:

Theorem 2. *[53] Let $x_1, x_2, \dots, x_n$ be n independent random variables, and $x_i \in [0,1]$ for $1 \le i \le n$. Denote $\mu = E\left[\sum_{i=1}^n x_i\right]$, then for an an arbitrary positive ϵ we have*

$$\Pr\left[\sum_{i=1}^{n} x_i \ge (1+\epsilon)\mu\right] \le e^{\frac{-\epsilon^2\mu}{2+\epsilon}} \tag{14}$$

Theorem 3. *Let $A_1, A_2, \dots, A_n$ be n events with happening probability $\Pr[A_1], \Pr[A_2], \dots, \Pr[A_n]$, then $\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] \le \sum_{i=1}^n \Pr[A_i]$.*

Moreover, α is used to ensure that the following defined expected values are fractional number and defined as follows:

$$\alpha = \min\left\{\min\left\{\frac{\widetilde{\lambda}\min(c(l))}{r.\delta}\right\}, \min\left\{\frac{r.D}{T(p_k)}\right\}, \min\left\{\frac{\pi(n)}{\eta(f)}\right\}\right\}$$
$$\forall r \in R, l \in \mathcal{L}, n \in \mathcal{N}, f \in \mathbb{F}, p_k \in P^{u,v} : u, v \in \mathcal{N} \tag{15}$$

where $\widetilde{\lambda}$ is the lower bound of maximum link load ratio value returned by the LP in Eq. (13). The proposed RRVA rounds the fractional $\widetilde{Y^{r,i,j}_{u,v,k}}$ solved from LP in Eq. (13) to integer value and then derive the routing and placement[4] solution. In the following, we will analyze the violating factor in terms of link capacity, requested delay and node capacity of RRVA.

5.3.1 *Link Capacity Violating Factor*

Definition 3. *For each request r and each link l, the traffic load z_l^r is defined as follows:*

$$z_l^r = \begin{cases} tr.\delta \text{ with prob.} \sum_{r\in R}\sum_{B\in\mathbb{B}_r}\sum_{f_i,f_j\in B}\sum_{u,v\in\mathcal{N}}\sum_{p_k\in P^{u,v}} \widetilde{Y^{r,i,j}_{u,v,k}} \cdot H^{u,v}_{l,k} \\ 0 \text{ otherwise.} \end{cases}$$

4. According to Eq. (13), $X_u^{r,f_i} \ge \sum_{p_k\in P^{u,v}} Y^{r,i,j}_{u,v,k}$, so the VNF placement solution achieved by RRVA can also be regarded as rounded from X_u^{r,f_i}.

Since $z_l^{r_1}$, $z_l^{r_2}$, ... are mutually independent according to their definition, the expected load on link l is

$$E\left[\sum_{r\in R} z_l^r\right] = \sum_{r\in R} E[z_l^r] = \sum_{r\in R} r.\delta \cdot \sum_{f_i,f_j}\sum_{u,v\in\mathcal{N}} \widetilde{Y^{r,i,j}_{u,v,k}} \cdot H^{u,v}_{l,k}$$
$$\le \widetilde{\lambda} \cdot c(l) \tag{16}$$

According to the definition of α in Eq. (15), it holds that $0 \le \frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)} \le 1$. Therefore, by dividing Eq. (16) with $\frac{\widetilde{\lambda}\cdot c(l)}{\alpha}$ on both sides we have

$$\mu_c = E\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)}\right] \le \alpha \tag{17}$$

Since $\frac{\alpha}{\mu_c} \ge 1$, we have

$$\Pr\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)} \ge (1+\epsilon)\alpha\right] \le \Pr\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)} \ge (1+\epsilon)\alpha\right] \tag{18}$$

We cannot directly apply Theorem 2 for $\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)}$ with α, however the following holds:

$$\alpha = \frac{\alpha}{\mu_c}\mu_c = \frac{\alpha}{\mu_c}E\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)}\right] = E\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)}\right] \tag{19}$$

By applying Theorem 2 for $\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)}$, based on Eq. (19)

$$\Pr\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{\widetilde{\lambda}\cdot c(l)} \ge (1+\epsilon)\alpha\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \tag{20}$$

where ϵ is an arbitrary positive value. Further, using inequalities from (18) and (20) together and introducing Δ, we have

$$\Pr\left[\sum_{r\in R}\frac{z_l^r}{\widetilde{\lambda}\cdot c(l)} \ge (1+\epsilon)\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \le \frac{\Delta}{N^2} \tag{21}$$

where Δ is a network related variables and $\Delta \to 0$ when the network size grows. By solving Eq. (21), we have that

$$\epsilon \ge \frac{-\log\frac{\Delta}{N} + \sqrt{\log^2\frac{\Delta}{N^2} - 8\alpha\log\frac{\Delta}{N^2}}}{2\alpha} \tag{22}$$

Theorem 4. *RRVA can achieve a link capacity violating factor of $\frac{4\log N}{\alpha} + 3$.*

Proof. By setting $\Delta = \frac{1}{N^2}$, Eq. (21) becomes

$$\Pr\left[\sum_{r\in R}\frac{z_l^r}{\widetilde{\lambda}\cdot c(l)} \ge (1+\epsilon)\right] \le \frac{1}{N^4}, \quad \text{where } \epsilon \ge \frac{4\log N}{\alpha} + 2. \tag{23}$$

By using Union Bound inequality in Theorem 3 for all the links

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

$$\Pr\left[\bigcup_{l\in\mathcal{L}}\sum_{r\in R}\frac{z_l^r}{\widetilde{\lambda}\cdot c(l)}\geq(1+\epsilon)\right]\leq\sum_{l\in\mathcal{L}}\Pr\left[\frac{z_l^r}{\widetilde{\lambda}\cdot c(l)}\geq(1+\epsilon)\right]$$
$$\leq N^2\cdot\frac{1}{N^4}=\frac{1}{N^2},\quad\text{where }\epsilon\geq\frac{4\log N}{\alpha}+2. \tag{24}$$

The last inequality holds since there are at most N^2 links in a network with N nodes. Finally, Eq. (24) indicates that the probability the expected load on any link violates $\widetilde{\lambda}\cdot c(l)$ with a factor of $1+\epsilon=\frac{4\log N}{\alpha}+3$ approaches 0 when $N\rightarrow+\infty$. □

Theorem 5. *RRVA can achieve a delay violating factor of* $\frac{\log|R||B_x|+2\log N}{\alpha}+3$.

Proof. For brevity, we have proved it in Appendix A. □

Theorem 6. *The randomized approximation algorithm can achieve a node capacity violating factor of* $\frac{3\log N}{\alpha}+3$.

Proof. For brevity, we have proved it in Appendix B. □

Theorem 7. *The randomized approximation algorithm can achieve a link bandwidth violating factor of* $\frac{4\log N}{\alpha}+3$, *delay violating factor of* $\frac{\log|R||B_x|+2\log N}{\alpha}+3$, *and node capacity violating factor of* $\frac{3\log N}{\alpha}+3$.

Proof. The proof follows from Theorems 4, 5 and 6. □

6 Simulations

6.1 Simulation Setup

We conduct simulations[5] on two networks: USANet, displayed in Fig. 4, which is a realistic carrier backbone network, and GÉANT, shown in Fig. 5, which is a pan-European communications infrastructure. In both networks, the solid red circled nodes are assumed to be edge clouds, whose capacities range from 15 to 20 *units*, and the dashed red circled nodes are assumed to be the public cloud nodes, whose capacities are infinite. For each link, its capacity is randomly picked in [3, 4] *Gb/s* and its delay takes value in [10,50] *ms*. It is assumed that there are in total 15 VNFs whose capacities vary from 5 to 10 *units*, and the node processing time for each VNF varies from 50 to 150 *ms* [57]. We randomly generate 50 sets of $|R|=20,40,60,80,100$ requests for requested both totally ordered SFC (the request set is denoted by R^t) and partially ordered SFC (the request set denoted by R^p), separately. R^t and R^p are uniformly distributed in each edge cloud. For each request $r(E,\delta,F,D)\in R^t$, δ is between 10 and 50 *Mb/s*, $|F|$ is between 5 and 10, and D is between 500 and 800 *ms*. The traffic request setting is in line with [58], and reflects the conventional application of e.g., Web Service, VoIP, etc. The request $r(E,\delta,F,D)\in R^p$ is generated the same with the request in R^t except that we set 4 segments of VNFs in each requested partially ordered SFC (the requested VNFs in R^p are identical with the ones in R^t for each request), and therefore the number of VNFs in each segment is in [1,3].

5. The simulations were run on a high-performance desktop PC with 3.40 GHz and 16 GB memory. We used IBM ILOG CPLEX (CPLEX Callable Library interface) 12.6 to implement OPT_LB, IBM ILOG CPLEX 12.6 together with C# to implement RRVA, and C# to implement the heuristic algorithms.

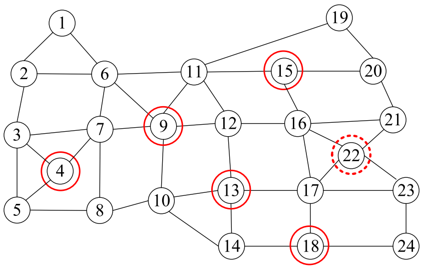


Fig. 4. USA carrier backbone network.

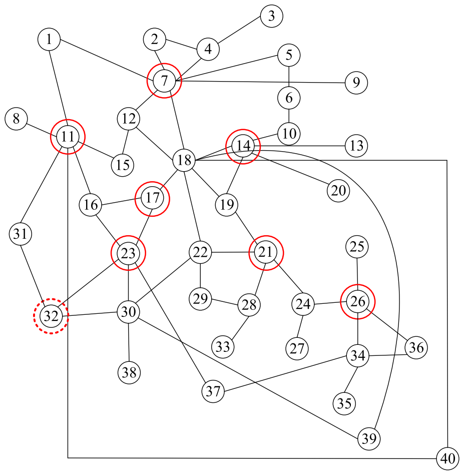


Fig. 5. GÉANT pan-European research network.

The solution of the LP from Eq. (13) provides a lower bound of the optimal solution (denoted by OPT_LB), so we compare OPT_LB and our proposed approximation algorithm RRVA with the relevant (slightly modeified) SFC-eMbedding APproach (MAP) [37] and two straightforward heuristics as follows. We initialize $i=1$.

1) SFC-MAP: For each request $r(E,\delta,F,D)$, suppose that f_0 is placed on $r.E$ (also for the Greedy and Random algorithm in below) as in segment m_0. For each totally ordered sub-SFC m of r, SFC-MAP first constructs a multi-layer graph by creating $|F|+1$ copies of the original graph. The inter-layer links are created to connect the same node in different layers, and the inter-layer nodes are created to represent the possible places to host each VNF in each layer. By assigning link delay weights, SFC-MAP runs shortest path from the ingress node in layer 1 to the egress node in layer $|F|+1$. If the solution is not found or the delay requirement is violated, a penalty is imposed for each link weight. The above procedure continues at most τ times, where we set $\tau=30$ in our simulation.
2) Greedy Algorithm: For each request $r(E,\delta,F,D)$, suppose that f_0 is placed on $r.E$ as in segment m_0. For each requested VNF f in segment m_i, Greedy algorithm first tries to place it on one of the nodes where the VNF(s) in segment m_{i-1} is hosted (denoted by the set N_{f_x}). If this step fails, Greedy algorithm places f on a node n_{f_y} with sufficient capacity, such that the $\frac{\min_{l\in p_k}(c(l)-b(l))}{T(p_k)}$ value is maximum among all the nodes $E\backslash\{N_{f_x}\}$, where

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

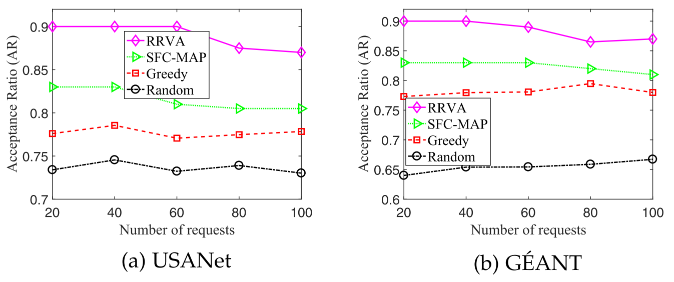


Fig. 6. Acceptance Ratio for the totally ordered SFC.

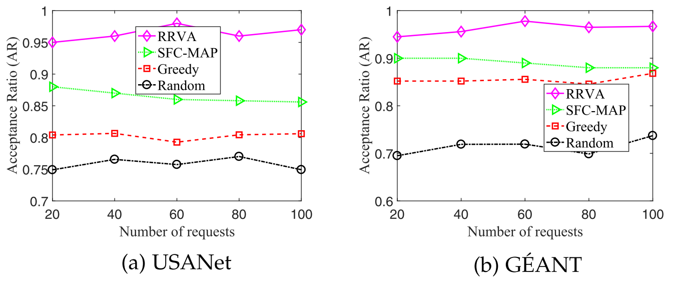


Fig. 7. Acceptance Ratio for the partially ordered SFC.

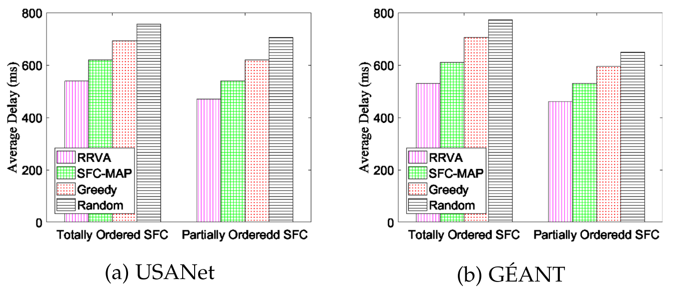


Fig. 8. Average Delay of $|R| = 100$ requests.

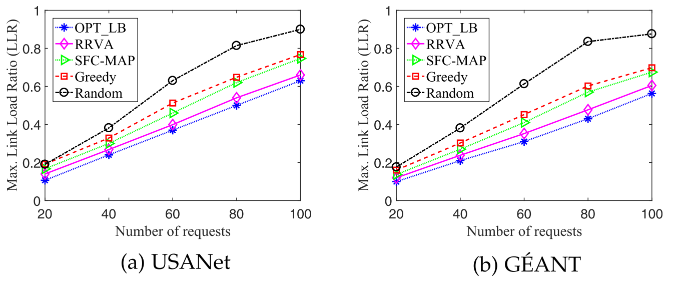


Fig. 9. Maximum Link Load Ratio (LLR) for the totally ordered SFC.

$\min_{l \in p_k}(c(l) - b(l))$ denotes the minimum link residual capacity in path p_k from n_{f_y} to each node in N_{f_x}. Moreover, the sum of each path's delay and node processing delay for f together with the delay of serving previous VNFs should be less than D.

3) Random Algorithm: For each requested VNF f in segment m_i, Random algorithm first tries to place it on one of the node(s) where the VNFs in segment m_{i-1} are hosted (denoted by N'_{f_x}). If this step fails, it randomly chooses one node n'_{f_y} with sufficient capacity such that there exists at least one path from each node in N'_{f_x} to n'_{f_y} whose accumulating delay is less than D and the path bandwidth is no less than δ. If succeeds, it randomly selects one satisfied path from each node in N'_{f_x} to n'_{f_y}.

For both Greedy and Random algorithms, the above procedures continue by increasing i by 1 until all the requested VNFs have been placed and associated paths have been found without violating delay, node capacity and link capacity constraints, otherwise the algorithms continue to serve the next request using the same routine until all the requests have been iterated.

6.2 Simulation Results

We first compare the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted requests divided by the total number of requests. Here, if a request is accepted, it means that its specified total flow traversing delay D is obeyed returned by the found solution. From the simulations we found that the variance of AR and maximum Link Load Ratio (LLR) for all the algorithms is small. To not clutter the figures, we only present the mean value of AR and Max. LLR in below. Moreover, since Opt_LB only returns the fractional solutions, we do not know the feasible integer solutions corresponding to different requests and thus cannot calculate its returned AR value. We therefore omit the AR values for Opt_LB. From Figs. 6 and 7 we can see that our approximation algorithm can accept almost all the requests (above around 90 percent) for both totally ordered SFC and partially ordered SFC. The AR values returned by Greedy algorithm and Random algorithm are much lower than the AR value achieved by RRVA. SFC-MAP has a better performance than Greedy and Random, but its achieved AR value is still lower than RRVA. For all the algorithms, their achieved AR value for the partially ordered SFC request in Fig. 7 is higher than the AR value for the partially ordered SFC request in Fig. 6. The reason is that as we stated in Section 3.1, the traversing delay in partially ordered SFC is lower than the delay in the totally ordered SFC, which in turns satisfy more requests regarding delay requirement. In particular, Fig. 8 illustrates the average delay value returned from all the algorithms among 100 requests. Accordingly, RRVA achieves the least average delay, followed SFC-MAP and Greedy, and the Random has the largest average delay.

Next, we evaluate the maximum Link Load Ratio (LLR) of all the algorithms. Figs. 9 and 10 show the Max. LLR value for all the algorithms. We find that Opt_LB achieves the lowest Max. LLR value, and the approximation algorithm achieves a very close performance with OPT_LB. SFC-MAP performs ranks the third regarding the performance of Max. LLR value. Random algorithm performs poorly by having the highest Max. LLR value because of its randomness in choosing placed node and paths regardless of link residual bandwidth. For all the algorithms, we found that their achieved Max. LLR value for the partially ordered SFC request in Fig. 10 is larger than the Max. LLR value for the partially ordered SFC request in Fig. 9. This is because more paths need to be established in the partially ordered SFC, and this will consume more bandwidth which results in larger Max. LLR.

After that, we verify the performances of the algorithms when different K (number of paths between node pairs) is set. As stated above, we omit the AR values for OPT_LB.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

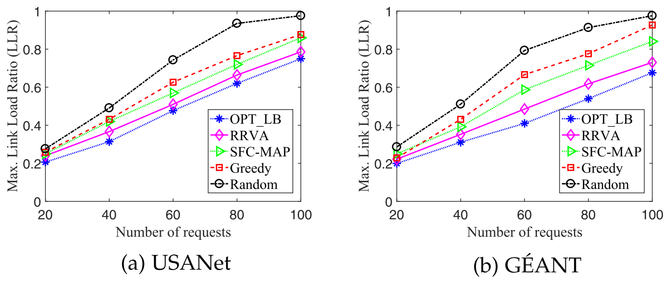


Fig. 10. Maximum Link Load Ratio (LLR) for the partially ordered SFC.

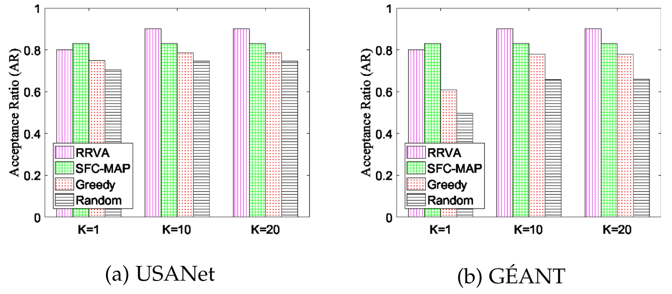


Fig. 11. Acceptance Ratio for K paths and totally ordered SFC ($|R| = 40$).

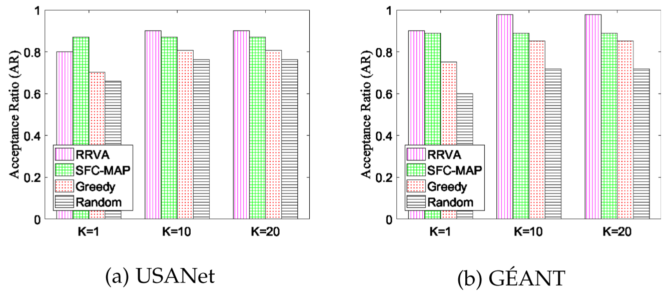


Fig. 12. Acceptance Ratio for K paths and partially ordered SFC ($|R| = 40$).

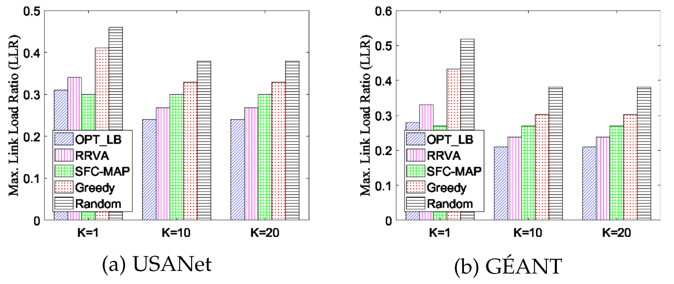


Fig. 13. Maximum Link Load Ratio (LLR) for K paths and totally ordered SFC ($|R| = 40$).

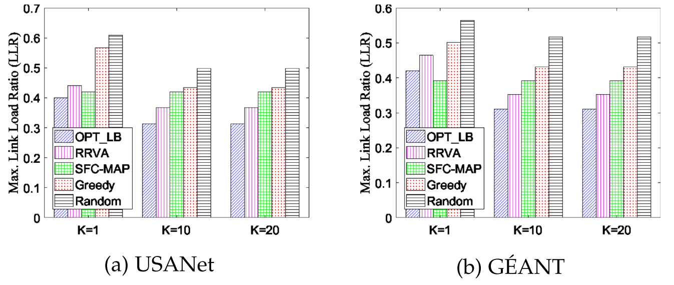


Fig. 14. Maximum Link Load Ratio (LLR) for K paths and partially ordered SFC ($|R| = 40$).

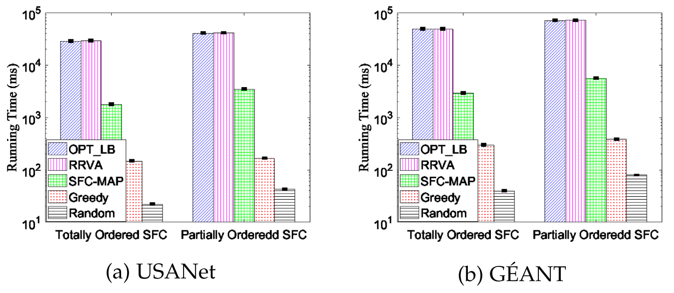


Fig. 15. Total Running Time for $|R| = 40$ requests (95 percent confidence interval).

For simplicity we only present the results when $|R| = 40$. We see that when K increases from 1 to 10, the AR value increases for all the algorithms except for SFC-MAP in Figs. 11 and 12, and the Max. LLR value decreases for all the algorithms in Figs. 13 and 14. But this trend does not change from 10 to 20 for both AR and Max. LLR, which implies that $K = 10$ paths are enough to guarantee QoS (e.g., delay). This is because SFC-MAP only applies shortest path algorithm to find route between VNF pair and therefore the number of paths does not affect its performance.

Finally, Fig. 15 presents the total running time of all the algorithms (in log scale) for $|R| = 40$, where the confidence interval is set to 95 percent. We see that the confidence intervals of all the algorithms are not very visible,[6] which indicates that the variances are also very small. We see that the OPT_LB has similar running time with RRVA since their time complexities are dominated by the LP, but both of them are higher than SFC-MAP and two other heuristics. Nevertheless, it is still acceptable since RRVA can achieve a close-to-optimal performance as shown in Figs. 6, 7, 8, 9, and 10. SFC-MAP has much higher running time than Greedy and Random heuristic due to its increased problem size ($|F| + 1$ times nodes and links) and maximum iteration times. Another observation is that when $|R| = 1$, we found our proposed approximation algorithm can return the solution in less than several seconds (We omit the figure for brevity). In practice, in many cases the traffic requests can be known or predicted [59] in advance, therefore the service provider can apply the proposed approximation algorithm to calculate the solutions in an offline fashion (in acceptably short time as implied in Fig. 15) for the (delay-sensitive) incoming traffic requests. In addition, our proposed approximation algorithm can also be applied only periodically or as complementary solution for existing methods. Moreover, for the same algorithm, we found that its running time for the totally ordered SFC is lower than the value for the partially ordered SFC. This can be explained that since the partially ordered SFC consists of one or more totally ordered sub-SFC, then there are more data input or constraints in the OPT_LB and RRVA, which results in higher running time. Similarly, Greedy and Random needs to establish more paths for VNF pairs in different segments, which also consumes more time.

6. Fig. 15 is log-scale that additionally contributes to the confidence interval visibility.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

7 Conclusion

In this paper, we have studied the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds, which is to place each end user's requested VNFs on network nodes and find routes among each adjacent VNF pair such that the maximum link load ratio λ is minimized and the total traversing delay is no greater than the specified value. We have considered this problem for both the totally ordered SFC and partially ordered SFC. We have proved that the DVPR problem is NP-hard, even for the special case when only one VNF is requested. We subsequently propose a randomized rounding approximation algorithm to solve it. Simulation results reveal that the proposed randomized rounding approximation algorithm outperforms three other heuristics and achieves close-to-optimal performances in terms of acceptance ratio and maximum link load ratio. In practice, the proposed approach can provide the service provider with the VNF placement and routing solutions for end users' requests such that each user's NFV delay requirement is satisfied while the maximum link load ratio is minimized. By minimizing the maximum link load ratio, the actual network flow is evenly distributed across the network, and more future requests can be accommodated since we try our best to reduce the network bottleneck. In our future work, we plan to implement our proposed algorithm in real testbeds or simulators and extend our work to further optimize network management and control related metric such as cross traffic management and network control overhead. Another improvement is to further shorten the running time of RRVA by e.g., reducing the feasible LP region [60].

Appendix A
User Delay Violating Factor

Proof. Similar to the proof of theorem 4, we have the following definition: □

Definition 4. *For each request $r \in R$, $B \in \mathbb{B}_r$, $f_i, f_j \in B$*

$$w^r_{f_i,f_j} = \begin{cases} T(p_k) + \frac{\Phi^{f_i}_u + \Phi^{f_j}_v}{2} \text{ with probability } \sum_{p_k \in Pu,v} \widetilde{Y^{r,i,j}_{u,v,k}} \\ 0 \text{ otherwise} \end{cases}.$$

Since $w^r_{f_0,f_1}, w^r_{f_1,f_2}, \ldots$are mutually independent according to their definition, the expected delay for r follows:

$$\begin{aligned} E\left[\sum_{f_i,f_j\in B} w^r_{f_i,f_j}\right] &= \sum_{f_i,f_j\in B} E[w^r_{f_i,f_j}] \\ &= \sum_{f_i,f_j\in B}\left(\sum_{p_k\in P^{u,v}} \widetilde{Y^{r,i,j}_{u,v,k}} \cdot \left(T(p_k) + \frac{\Phi^{f_i}_u + \Phi^{f_j}_v}{2}\right)\right) \le D \end{aligned} \tag{25}$$

According to the definition of α in Eq. (15) it holds that $0 \le \frac{w^r_{f_i,f_j}\cdot\alpha}{D} \le 1$. By dividing Eq. (25) with $\frac{D}{\alpha}$ on both sides we have

$$\mu_D = E\left[\sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}\cdot\alpha}{D}\right] \le \alpha \tag{26}$$

In a similar way as proved from Eqs. (17), (18), (19), (20), and (21), using $\frac{\alpha}{\mu_D} \ge 1$ (Eq. (26)) and applying Theorem 2 for $\frac{\alpha}{\mu_D}\frac{w^r_{f_i,f_j}\cdot\alpha}{D}$ whose expectation of their sum over $f_i, f_j \in r$ is α, we arrive at

$$\begin{aligned} &\Pr\left[\sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}\cdot\alpha}{D} \ge (1+\theta)\alpha\right] \le \\ &\Pr\left[\sum_{f_i,f_j\in B} \frac{\alpha}{\mu_D}\frac{w^r_{f_i,f_j}\cdot\alpha}{D} \ge (1+\theta)\alpha\right] \le e^{\frac{-\theta^2\alpha}{2+\theta}} \end{aligned} \tag{27}$$

By letting $e^{\frac{-\theta^2\alpha}{2+\theta}}$ be less than $\frac{\Delta}{|R||B_x|}$, we have

$$\Pr\left[\sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}}{D} \ge (1+\theta)\right] \le e^{\frac{-\theta^2\alpha}{2+\theta}} \le \frac{\Delta}{|R||B_x|} \tag{28}$$

where $|B_x|$ is the maximum number of totally ordered sub-SFCs among all the requests. Eq. (28) is satisfied for

$$\theta \ge \frac{-\log\frac{\Delta}{|R||B_x|} + \sqrt{\log^2\frac{\Delta}{|R||B_x|} - 8\alpha\log\frac{\Delta}{|R||B_x|}}}{2\alpha} \tag{29}$$

By setting $\Delta = \frac{1}{N^2}$, Eq. (28) becomes

$$\Pr\left[\sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}}{D} \ge (1+\theta)\right] \le \frac{1}{|R||B_x|N^2} \tag{30}$$

Using Union Bound inequality in Theorem 3 for all the requests, we arrive at

$$\begin{aligned} &\Pr\left[\bigcup_{r\in R,B\in B_r} \sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}}{D} \ge (1+\theta)\right] \\ &\le \sum_{r\in R,B\in B_r} \Pr\left[\sum_{f_i,f_j\in B} \frac{w^r_{f_i,f_j}}{D} \ge (1+\theta)\right] \\ &\le |R||B_x| \cdot \frac{1}{|R||B_x|N^2} = \frac{1}{N^2}, \text{where } \theta \ge \frac{\log|R||B_x| + 2\log N}{\alpha} + 2. \end{aligned} \tag{31}$$

Eq. (31) indicates that for any request, the probability the expected delay violates the specified delay value with a factor of $1+\theta = \frac{\log|R||B_x|+2\log N}{\alpha} + 3$ approaches 0 when $N \to +\infty$.

Appendix B
Node Capacity Violating Factor

Proof. □

Definition 5. *For VNF f and each node n, the traffic load β^f_l is defined as follows:*

$$\beta^f_n = \begin{cases} \eta(f) \text{ with prob. } max_{r\in R:f\in r.F}\widetilde{X^{r,f}_n} \quad \forall n \in \mathcal{N}\backslash\mathcal{C} \\ 0 \text{ otherwise} \end{cases}.$$

Since $\beta^{f_1}_n, \beta^{f_2}_n, \ldots$are mutually independent according to their definition, the expected load on node n is

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

$$E\left[\sum_{f\in\mathcal{F}}\beta_n^f\right] = \sum_{f\in\mathcal{F}} E[\beta_n^f] = \sum_{f\in\mathcal{F}} \max_{r\in R: f\in r.F} \widetilde{X_n^{r,f}} \cdot \eta(f) \le \pi(n) \quad \forall n \in \mathcal{N}\backslash\mathcal{C} \tag{32}$$

According to the definition of α in Eqs. (15) and (32), it holds that $0 \le \frac{\beta_n^f\cdot\alpha}{\pi(n)} \le 1$. By dividing Eq. (32) with $\frac{\pi(n)}{\alpha}$ on both sides we have

$$\mu_\pi = E\left[\sum_{f\in\mathcal{F}}\frac{\beta_n^f\cdot\alpha}{\pi(n)}\right] \le \alpha \tag{33}$$

In a similar way as proved from Eqs. (17), (18), (19), (20), and (21), using $\frac{\alpha}{\mu_\pi} \ge 1$ (Eq. (33)) and applying Theorem 2 for $\frac{\alpha}{\mu_\pi}\frac{\beta_n^f\cdot\alpha}{\pi(n)}$ whose expectation of their sum over $f \in \mathcal{F}$ is α, we arrive at

$$\Pr\left[\sum_{f\in\mathcal{F}}\frac{\beta_n^f\cdot\alpha}{\pi(n)} \ge (1+\rho)\alpha\right] \le \Pr\left[\sum_{f\in\mathcal{F}}\frac{\alpha}{\mu_\pi}\frac{\beta_n^f\cdot\alpha}{\pi(n)} \ge (1+\rho)\alpha\right] \le e^{\frac{-\rho^2\alpha}{2+\rho}} \tag{34}$$

where ρ is an arbitrary positive value. Further, by letting $e^{\frac{-\rho^2\alpha}{2+\rho}}$ be less than $\frac{\Delta}{N}$, we arrive at

$$\Pr\left[\sum_{f\in\mathcal{F}}\frac{\beta_n^f}{\pi(n)} \ge (1+\rho)\right] \le e^{\frac{-\rho^2\alpha}{2+\rho}} \le \frac{\Delta}{N} \tag{35}$$

By solving Eq. (35), we have that

$$\rho \ge \frac{-\log\frac{\Delta}{N} + \sqrt{\log^2\frac{\Delta}{N^2} - 8\alpha\log\frac{\Delta}{N^2}}}{2\alpha} \tag{36}$$

By setting $\Delta = \frac{1}{N^2}$, Eq. (35) becomes $\Pr\left[\sum_{f\in\mathcal{F}}\frac{\beta_n^f\cdot\alpha}{\pi(n)} \ge (1+\rho)\alpha\right] \le \frac{1}{N^3}$

As a result, for all the nodes, we have

$$\begin{aligned}&\Pr\left[\bigcup_{n\in\mathcal{N}}\sum_{f\in\mathcal{F}}\frac{\beta_n^f}{\pi(n)} \ge (1+\rho)\right]\\ &\le \sum_{n\in\mathcal{N}}\Pr\left[\sum_{f\in\mathcal{F}}\frac{\beta_n^f}{\pi(n)} \ge (1+\rho)\right]\\ &\le N\cdot\frac{1}{N^3} = \frac{1}{N^2},\ \text{where } \rho \ge \frac{3\log N}{\alpha} + 2.\end{aligned} \tag{37}$$

Therefore, Eq. (37) implies that for any node in the network, the probability that its capacity will be violated by a factor of $1+\rho = \frac{3\log N}{\alpha} + 3$ will approach 0 when N grows $+\infty$.

ACKNOWLEDGMENTS

The work of Song Yang is partially supported by the National Natural Science Foundation of China (NSFC, No. 61802018) and Beijing Institute of Technology Research Fund Program for Young Scholars. The work of Fan Li is partially supported by the NSFC (No. 61772077, 61432015), and the Beijing Natural Science Foundation (No. 4192051). The work of Xu Chen is partially supported by the NSFC (No. U1711265, No. 61972432) and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No. 2017ZT07X355). The work of Xiaoming Fu is partially supported by EU FP7 CleanSky ITN (No. 607584) and H2020 RISE COSAFE (No. 824019) projects.

REFERENCES

[1] P. Mell and T. Grance, “The NIST definition of cloud computing,” *Commun. ACM*, vol. 53, no. 6, 2010, Art. no. 50.
[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing: A key technology towards 5G,” ETSI White Paper, 2015.
[3] ETSI Publishes First Specifications for Network Functions Virtualisation. [Online]. Available: http://www.etsi.org/news-events/news/700-2013-10-etsi-publishes-first-nfv-specifications, Accessed: 2019.
[4] R. Guerzoni, “Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action introductory white paper,” in *Proc. SDN OpenFlow World Congr.*, Jun. 2012, pp. 1–16.
[5] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, “Service function chaining in next generation networks: State of the art and research challenges,” *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 216–223, Feb. 2017.
[6] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, “Efficient algorithms for capacitated cloudlet placements,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.
[7] A. Ceselli, M. Premoli, and S. Secci, “Mobile edge cloud network design optimization,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, Jun. 2017.
[8] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, “Selective offloading in mobile edge computing for the green internet of things,” *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.
[9] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, “Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications,” *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 475–488, Mar. 2018.
[10] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Jul.–Sep. 2017.
[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.
[12] 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, Jan.–Mar. 2016.
[13] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, “A survey on service function chaining,” *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, 2016.
[14] B. Yi, X. Wang, K. Li, M. Huang, et al., “A comprehensive survey of network function virtualization,” *Comput. Netw.*, vol. 133, pp. 212–262, 2018.
[15] J. G. Herrera and J. F. Botero, “Resource allocation in NFV: A comprehensive survey,” *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.
[16] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *Proc. IEEE 3rd Int. Conf. Cloud Netw.*, 2014, pp. 7–13.
[17] 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.*, vol. 25, no. 4, pp. 2008–2025, Aug. 2017.
[18] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, “Near optimal placement of virtual network functions,” in *Proc. IEEE INFOCOM*, 2015, pp. 1346–1354.
[19] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, “Deploying chains of virtual network functions: On the relation between link and server usage,” in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

[20] L. Guo, J. Pang, and A. Walid, “Joint placement and routing of network function chains in data centers,” in *Proc. IEEE INFOCOM*, 2018, pp. 612–620.

[21] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, “A scalable approach for service chain mapping with multiple SC instances in a wide-area network,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.

[22] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, “On dynamic service function chain deployment and readjustment,” *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 543–553, Sep. 2017.

[23] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, “Optimal virtual network function placement in multi-cloud service function chaining architecture,” *Comput. Commun.*, vol. 102, pp. 1–16, 2017.

[24] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, and A. Erbad, “Multi-cloud distribution of virtual functions and dynamic service deployment: Open ADN perspective,” in *Proc. IEEE Int. Conf. Cloud Eng.*, 2015, pp. 299–304.

[25] H. Goudarzi and M. Pedram, “Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems,” in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 324–331.

[26] M. Su, L. Zhang, Y. Wu, K. Chen, and K. Li, “Systematic data placement optimization in multi-cloud storage for complex requirements,”*IEEE Trans. Comput.*, vol. 65, no. 6, pp. 1964–1977, 2016.

[27] F. L. Pires and B. Barán, “Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach,” in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput.*, 2013, pp. 203–210.

[28] K. Mills, J. Filliben, and C. Dabrowski, “Comparing VM-placement algorithms for on-demand clouds,” in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 91–98.

[29] 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, Sep. 2016.

[30] H. A. Alameddine, L. Qu, and C. Assi, “Scheduling service function chains for ultra-low latency network services,” in *Proc. 13th Int. Conf. Netw. Service Manage.*, 2017, pp. 1–9.

[31] Z. Zhang, Z. Li, C. Wu, and C. Huang, “A scalable and distributed approach for NFV service chain cost minimization,” in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 2151–2156.

[32] Q. Li, Y. Jiang, P. Duan, M. Xu, and X. Xiao, “Quokka: Latency-aware middlebox scheduling with dynamic resource allocation,” *J. Netw. Comput. Appl.*, vol. 78, pp. 253–266, 2017.

[33] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, “NFP: Enabling network function parallelism in NFV,” in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 43–56.

[34] N. Huin, B. Jaumard, and F. Giroire, “Optimal network service chain provisioning,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1320–1333, Jun. 2018.

[35] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, “Virtual function placement for service chaining with partial orders and anti-affinity rules,” *Netw.*, vol. 71, no. 2, pp. 97–106, 2018.

[36] A. Dwaraki and T. Wolf, “Adaptive service-chain routing for virtual network functions in software-defined networks,” in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization*, 2016, pp. 32–37.

[37] J. Pei, P. Hong, K. Xue, and D. Li, “Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Oct. 2019.

[38] S. Yang, F. Li, R. Yahyapour, and X. Fu, “Delay-sensitive and availability-aware virtual network function scheduling for NFV,” *IEEE Trans. Serv. Comput.*, early access, Jul. 9, 2019, doi: 10.1109/TSC.2019.2927339.

[39] I.-H. Hou, T. Zhao, S. Wang, and K. Chan, “Asymptotically optimal algorithm for online reconfiguration of edge-clouds,” in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2016, pp. 291–300.

[40] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, “Cost-efficient workload scheduling in cloud assisted mobile edge computing,” in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–10.

[41] N. Saha, S. Bera, and S. Misra, “Sway: Traffic-aware QoS routing in software-defined IoT,” *IEEE Trans. Emerging Topics Comput.*, early access, Jun. 14, 2018, doi: 10.1109/TETC.2018.2847296.

[42] S. Bera, S. Misra, and A. Jamalipour, “FlowStat: Adaptive flow-rule placement for per-flow statistics in SDN,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 530–539, Mar. 2019.

[43] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, “Dynamic, latency-optimal vNF placement at the network edge,” in *Proc. IEEE INFOCOM*, 2018, pp. 693–701.

[44] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, “Network function placement on virtualized cellular cores,” in *Proc. IEEE Int. Conf. Commun. Syst. Netw.*, 2017, pp. 259–266.

[45] A. Gupta, M. Tomatore, B. Jaumard, and B. Mukherjee, “Virtual-mobile-core placement for metro network,” in *Proc. IEEE Conf. Netw. Softwarization Workshops*, 2018, pp. 308–312.

[46] D. Bhamare, A. Erbad, R. Jain, M. Zolanvari, and M. Samaka, “Efficient virtual network function placement strategies for cloud radio access networks,” *Comput. Commun.*, vol. 127, pp. 50–60, 2018.

[47] M. Leconte, A. Destounis, and G. Paschos, “Traffic engineering with precomputed pathbooks,” in *Proc. IEEE INFOCOM*, 2018, pp. 234–242.

[48] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, “Traffic aware placement of interdependent NFV middleboxes,” in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.

[49] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.

[50] C.-L. Li, S. McCormick, and D. Simchi-Levi, “The complexity of finding two disjoint paths with min-max objective function,” *Discrete Appl. Math.*, vol. 26, no. 1, pp. 105–115, 1990.

[51] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, “Nonlinear integer programming,” in *50 Years of Integer Programming 1958–2008*. Berlin, Germany: Springer, 2010, pp. 561–618.

[52] K. M. Anstreicher, “Linear programming in O ([n3/ln n]L) operations,” *SIAM J. Optimization*, vol. 9, no. 4, pp. 803–812, 1999.

[53] R. Tarjan, *Course: Advanced Algorithm Design. Lecture: Chernoff, Probability and Computing*. Princeton, NJ, USA: Princeton Univ., 2009.

[54] L. Comtet, *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. Dordrecht, the Netherlands: Springer, 1974.

[55] H. Xu, Z. Yu, X. Y. Li, L. Huang, C. Qian, and T. Jung, “Joint route selection and update scheduling for low-latency update in SDNs,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3073–3087, Oct. 2017.

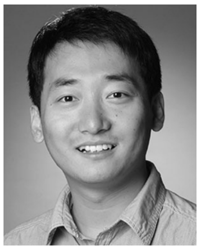
[56] S.-C. Lin, P. Wang, I. Akyildiz, and M. Luo, “Towards optimal network planning for software-defined networks,” *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2953–2967, Dec. 2018.

[57] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, “Elastic virtual network function placement,” in *Proc. IEEE 4th Int. Conf. Cloud Netw.*, 2015, pp. 255–260.

[58] M. Savi, M. Tornatore, and G. Verticale, “Impact of processing costs on service chain placement in network functions virtualization,” in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2015, pp. 191–197.

[59] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, “Improving traffic forecasting for 5G core network scalability: A machine learning approach,” *IEEE Netw.*, vol. 32, no. 6, pp. 42–49, Nov./Dec. 2018.

[60] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*. Berlin, Germany: Springer, 2001.



Song Yang received the PhD degree from the Delft University of Technology, the Netherlands, in 2015. From August 2015 to July 2017, he worked as a postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an associate professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests focus on data communication networks, cloud/edge computing, and network function virtualization. He is a member of the IEEE and ACM.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.

Fan Li received the BEng and MEng degrees in communications and information system from the Huazhong University of Science and Technology, China, in 1998 and 2001, respectively, the MEng degree in electrical engineering from the University of Delaware, in 2004, and the PhD degree in computer science from the University of North Carolina at Charlotte, in 2008. She is currently a professor with the School of Computer Science, Beijing Institute of Technology, China. Her current research focuses on wireless networks, ad hoc and sensor networks, and mobile computing. Her papers won Best Paper Awards from IEEE MASS (2013), IEEE IPCCC (2013), ACM MobiHoc (2014), and Tsinghua Science and Technology (2015). She is a member of the IEEE and ACM.

Stojan Trajanovski received the master's (with distinction) degree in advanced computer science from the University of Cambridge, United Kingdom, in 2011, and the PhD (cum laude) degree from the Delft University of Technology, The Netherlands, in 2014. He is currently a research scientist in Philips Research in Eindhoven, The Netherlands. Before that, he spent some time as a postdoctoral researcher with the University of Amsterdam and with the Delft University of Technology. He successfully participated at international science olympiads, winning a bronze medal at the International Mathematical Olympiad (IMO) in 2003. His main research interests include network science & complex networks, machine learning, game theory, and optimization algorithms. He is a member of the IEEE.

Xu Chen received the PhD degree in information engineering from the Chinese University of Hong Kong, in 2012. He is a full professor with Sun Yat-sen University, Guangzhou, China, and the vice director of the National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He worked as a postdoctoral research associate with Arizona State University, Tempe from 2012 to 2014, and a Humboldt scholar fellow with the Institute of Computer Science, University of Goettingen, Germany, from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, Honorable Mention Award of 2010 IEEE International Conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an area editor of the *IEEE Open Journal of the Communications Society*, an associate editor of the *IEEE Transactions on Wireless Communications*, the *IEEE Internet of Things Journal,* and the *IEEE Journal on Selected Areas in Communications (JSAC) Series on Network Softwarization and Enablers*. He is a member of the IEEE.

Yu Wang received the BEng and MEng degrees from Tsinghua University, and the PhD degree from the Illinois Institute of Technology, all in computer science. He is currently a professor with the Department of Computer and Information Sciences, Temple University. Prior to joining Temple University, he was a professor of computer science with the University of North Carolina at Charlotte (UNC Charlotte). His research interests include wireless networks, smart sensing, and mobile computing. His research has been continuously supported by US National Science Foundation and US Department of Transportation. He has published more than 200 papers in peer reviewed journals and conferences, with four best paper awards. He has served as general chair, program chair, program committee member, etc., for many international conferences (such as IEEE IPCCC, ACM MobiHoc, IEEE INFOCOM, IEEE GLOBECOM, and IEEE ICC). He has served as an editorial board member of several international journals, including the *IEEE Transactions on Parallel and Distributed Systems*. He is a recipient of Ralph E. Powe Junior Faculty Enhancement Awards from Oak Ridge Associated Universities, 2006, Outstanding Faculty Research Award from the College of Computing and Informatics, UNC Charlotte, 2008, and a fellow of the IEEE, 2018. He is also a senior member of the ACM and a member of the AAAS.

Xiaoming Fu received the PhD degree in computer science from Tsinghua University, Beijing, China, in 2000. He was then a research staff with the Technical University Berlin until joining the University of Göttingen, Germany, in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. He has spent research visits with the Universities of Cambridge, Uppsala, UPMC, Columbia, UCLA, Tsinghua, Nanjing, Fudan, and PolyU of Hong Kong. His research interests include network architectures, protocols, and applications. He is currently an editorial board member of the *IEEE Communications Magazine*, the *IEEE Transactions on Network and Service Management*, and the *Elsevier Computer Communications*, and has served on the organization or program committees of leading conferences such as INFOCOM, ICNP, ICDCS, MOBICOM, MOBIHOC, CoNEXT, ICN, and COSN. He is a senior member of the IEEE, an IEEE Communications Society distinguished lecturer, a fellow of the IET, and member of the Academia Europaea.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on April 07,2023 at 02:40:47 UTC from IEEE Xplore. Restrictions apply.