Latency-Efficient VNF Deployment and Path Routing for Reliable Service Chain

Yang Chen and Jie Wu, *Fellow, IEEE*

***Abstract*—Network services usually chain multiple types of Virtual Network Functions (VNFs) together in a specific order, known as service chain. One important issue of providing network services is reliability, which means that each type of VNF in a service chain acts properly on its function. The effective way to guarantee the reliability is to utilize sufficient redundancy through providing VNF backup instances beyond active primary ones. Software Defined Networking (SDN) enables the dedicate location pickup of primary and backup instances on switch-connected servers. The deployment of primary and backup VNF instances plays an important role of flow routing because of its influence on the transmission latency of the flow. In order to minimize the total latency of flows, we first formulate the reliable service chain deployment as a mathematical optimization problem. A detailed analysis of both software and hardware failure models is studied. Then we propose optimal solutions for the case of a single flow. Next, we prove the NP-hardness of our problem for the case with multiple flows and propose a heuristic strategy. Additionally, a performance-guaranteed solution is included if the server capacity is infinite. Extensive simulations are conducted to evaluate our proposed solutions compared to multiple algorithms.**

***Index Terms*—Backup, latency, reliability, service chain, VNF.**

I. Introduction

SOFTWARE Defined Networking (SDN) emerged in recent years to fundamentally change how we design, build and manage networks [1], which promotes the development of Network Function Virtualization (NFV) [2], [3]. NFV has been proposed to transform the implementation of network functions from expensive hardwares to software middleboxes, called Virtual Network Functions (VNFs) [4]. Network services usually chain multiple types of VNFs together in a specific order, known as service chain [5]. VNFs are most commonly provisioned in modern networks, demonstrating their increasing importance [6]. Additionally, the technical combination of SDN and NFV enables network service providers to pick VNFs' locations from multiple available servers and maneuvers traffic through appropriate VNFs [7]. However, VNFs are executed on virtualization platforms, which makes them more prone and more vulnerable to error compared with the expensive and dedicated hardware [8]. Reliability is an important requirement for network operators when offering specific services (e.g., voice call and video on demand), no matter through physical or virtual network appliances [9]. Carriers need to guarantee that service reliability (also known service chain resilience) and service level agreement are not affected when evolving to NFV [10]. Note that some purpose-built networks with regular and rich topologies [11] can provide the traditional five-nines reliability, but cannot be extended to general-purpose networks where NFV is commonly applied.

The effective way to guarantee the reliability is to utilize sufficient redundancy through providing VNF backup instances beyond active primary ones. Backups will be activated when their primary VNF instances fail [12]. The reliability of service provisioning may require the consolidation and migration of VNFs based on traffic load and user demand. All these operations create new points of failure that should be handled automatically [13]. SDN enables the dedicate location pickup of primary and backup instances on switch-connected servers. The deployment of primary and backup VNF instances plays an important role of flow routing because of its influence on the transmission latency of the flow. If primary and backup instances are not deployed efficiently on servers, the network performance may be greatly affected, including highly prolonged transmission latency and dramatically increased changes in forwarding rules that are stored in routers because more rules are needed to reroute flows.With the probabilistic prior failure information of hardware severs and software VNFs [4], we consider the joint problem of reliable VNF deployment and flow routing with an objective of minimizing the total transmission latency of all flows by considering failures and subsequent reroutes of flows [5]. The problem with only VNF deployment is already extremely challenging, which has been proven to be NP-hard under various types of objectives [7], [14], [15]. Taking consideration of the reliability issue, there is always a need to meet a certain level of service chain availability, which further complicates our problem. This is because we need to select and deploy backup VNF instances to increase the availability of the provided service chain. Additionally, various types of VNF active and backup instances compete for server resources. What's more, when an active VNF instance fails, a flow needs to reroute to the backup instance of the function to get processed, which

Manuscript received August 1, 2020; revised November 28, 2020; accepted December 26, 2020. Date of publication December 30, 2020; date of current version March 17, 2021. This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, and CNS 1564128. Recommended for acceptance by Dr. Di Wu. *(Corresponding author: Yang Chen.)*

The authors are with the Center for Networked Computing in Temple University, Philadelphia, Pennsylvania USA (e-mail: yang.chen@temple.edu; jiewu@temple.edu).

Digital Object Identifier 10.1109/TNSE.2020.3048033

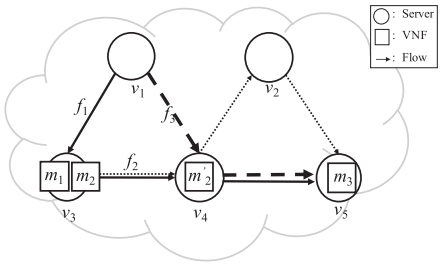


Fig. 1. Motivating example.

adds extra transmission latency. Therefore, the routing path of a flow also needs to be paid extra attention.

In this paper, we aim at minimizing the total latency of all flows given the failure probability information of servers and VNFs. We first formulate the reliable service chain deployment as a mathematical optimization problem. A detailed analysis of both software and hardware failure models is studied. Then we propose solutions for our problem. We start with the case of a single flow by introducing two optimal solutions for the homogeneous and heterogeneous VNFs with the same or different configurations. Next, we prove the NP-hardness of our problem for the case with multiple flows and propose an intuitive strategy. Additionally, a performance-guaranteed solution is included with an approximation ratio if the server capacity is infinite.

Here we use a motivating example in Fig. 1 to illustrate the challenges of our problem. We use a shaded square to denote a VNF instance. There are five switch-connected servers (denoted as cycle nodes), $v_1 \sim v_5$, where VNFs including the primary and backup instances can be deployed. Each server has a capacity, which is the total amount of resources that can be used to deploy VNFs. We omit the link between every pair of servers. Each link has a bandwidth capacity as well. There are three flows f_1, f_2, and f_3, whose sources and destinations are shown in Fig. 1. We use different lines to illustrate different flows and the link thickness to denote the flow rate. For example, f_3 has the largest rate, denoted with the thickest dashed line. We use squares to denote VNF instances. Each type of VNF instance has a distinct availability and each service chain has a requirement of reliability. We deploy backup instances in order to increase the service chain availability, where the selection of the number and the location of VNF instances plays an important role. We assume the primary and backup instances of the same type of VNF consumes the same amount of server resource, and have the same availability irrelevant to their deployed servers.

We take flow f_1 as an example in Fig 1. Suppose it requires a service chain $m_1 \rightarrow m_2 \rightarrow m_3$. We use $\rightarrow$ to illustrate the sequence order among VNFs in a service chain. Its original path is $v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$. To meet the requirement of f_1's reliability, here we deploy a backup instance of VNF m_2 as m_2'. In order to eliminate the hardware failure of a server, we prefer to deploy the primary and backup instances on different servers. However, this incurs the rerouting issue when the primary VNF fails. If m_2 fails, we need to reroute f_1 to get processed by m_2'. If m_2' is deployed casually, it is highly possible to have a longer path with a larger transmission delay, for example, deploying m_2' on v_2. Then its routing path will be $v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2 \rightarrow v_5$. But if m_2' is deployed on v_4, its routing path can keep the same. Additionally, multiple flows compete for the limited link resource as well as server resource. For example, f_1 and f_3 with the same source and destination compete for the link between v_1 and v_4, while f_1 and f_2 compete for servers v_3 and v_4. All of the above factors, including flow routing, backup selection and deployment, and service chain requirement, complicate our problem, making it extremely challenging.

Our contributions of this paper are listed as follows:

1) We innovatively formulate the backup allocation and assignment as a latency optimization problem. A detailed analysis of both software and hardware failure models is studied as well. We simplify some constraints of the network model in order to generate inspiring theoretical results.
2) We introduce two efficient greedy solutions for the homogeneous and heterogeneous VNFs with the same or different configurations when there is only a single flow. We prove the optimality of our proposed solutions.
3) For multiple flows, we prove the NP-hardness of our problem and propose an intuitive strategy under the setting of the limited server capacity. Additionally, a performance-guaranteed solution is included with an approximation ratio if the server capacity is infinite.

The remainder of this paper is organized as follows. Section II surveys related works. We describe our network model and formulate the problem in Section III. A detailed analysis of both software and hardware failure models is studied in Section IV. Section V proposes the optimal solutions for the single service chain with homogeneous and heterogeneous VNFs. Section VI solves the multiple service chain resilience problem with performance-guaranteed solutions. Section VII includes our simulation part and Section VIII concludes the paper.

II. Related Work

In this section, we give a brief review of state-of-the-art works [14], [16]. Several studies consider the placement of a minimum number of VNF instances to cover all the flows. While the case of a single type of network function is considered in [17], the case of multiple types of network functions is addressed in [15]. The work of [18] considers the placement of middleboxes to keep the shortest path between communicating pairs below threshold, but does not consider multiple network functions. In other related domains, such as SDN and link cloud computing, similar problems have also been studied. For example, the work in [19] considers the placement of SDN-enabled routers to maximize the total processed traffic. They consider a total resource constraint but neglect the limited resource constraint. Similarly, in the work on link cloud computing [20], although the resource constraints are considered, their proposed solution is only for a special case, and the overall problem does not consider the

multi-dimensional setting. To the best of our knowledge, the setting has rarely been considered except in a limited number of studies. In [21], the authors consider multi-resource VNFs with a focus on the analysis of the vertical scaling (i.e., scaling up/down of various resources) and horizontal scaling (i.e., varying number of VNFs instances). The work of [22] focuses only on request admission and routing. In [23], although the multi-resource setting is considered, the focus is on how to balance the traffic load across servers, taking into account different resource requirements by different network functions.

Most of the existing work on resilience design for NFV is focused on the standby deployment model [12], [24], [25]. This model requires at least one standby VNF instance for each primary VNF instance so as to ensure certain availability when failure occurs. Instead, [24] considers the hot-standby resilience design, where each standby instance is also active and is consistently synchronized with the primary instance. In contrast to backing up instances of the same type of network functions, [25] takes into account the inherent resource-sharing property of the virtualized resources and considers models where each backup server can be provisioned for multiple types of VNFs, and the backup server can be up and running as a certain type of VNF when failure occurs at some instances of the corresponding network function. Taking a different path, [12] studies the problem of VNF recovery by dynamically reallocating the processing resource of VNF instances. Instead of having one primary instance and one backup instance, both instances are actively functioning and have their states synchronized with each other. When one instance fails, the other instance will dynamically adjust its processing resource to accommodate the traffic that is supposed to be processed by the faulty instance. Taking advantage of the cloud networks, [26] minimizes the backup cost with edge resource constraints when considering the availability requirements. It also studies both the static and the dynamic backup situation. All the above work only focuses on the VNF deployment with backups for each VNF in the service chain, which sometimes wastes too much resource for the backup instances. They do not consider the routing path. In this paper, we selectively deploy primary and backup instances in order to save server capacity as well as finding the routing paths for flows.

III. Network Model and Problem Formulation

In this section, we introduce our network model and formulate the joint NFV deployment and flow routing problem as a mathematical optimization problem.

A. Network Model

The network consists of some commodity servers and switches, which are wired up and serve as a data center. We are given the set of switch-connected server nodes by V, satisfying $|V| = n$, where $V = \{v_1, v_2, \ldots, v_n\}$. $|\cdot|$ is the cardinality of a set. Each server $v_i \in V$ has a limited resource capacity in terms of the total amount of resource to support VNF instances, denoted as c_i. q_i is the availability of server v_i. We define X_i as an indicator parameter. If server v_i is available, $X_i = 1$; otherwise, $X_i = 0$.

TABLE I
Symbols and Definitions

Symbols	Definitions
V, E, F, S	the set of vertices, edges, flows, and service chains
v_i, c_i	the i_{th} server node and its capacity of VNF instances
e_{ij}, b_{ij}, d_{ij}	edge between v_i and v_j and its bandwidth, delay
ϕ_{ij}	number of VNF m_j instances on v_i
q_i, p_{ij}	availability of server v_i and VNF m_j on v_i
f, d_f, r_f, p_f	a flow, its total transmission delay, traffic rate, path
s_f, ρ_s	f's service chain and requested availability
X_i, Y_{ij}	Indicator whether server v_i is available
Y_{ij}	Indicator whether VNF m_j on v_i is available

A flow f has a total upper-bound transmission delay of d_f, a traffic rate of r_f, and a path of p_f. The path set of all flows is denoted as $P = \{p_f | f \in F\}$. The given set of service chains is denoted as S. A flow f requested to be processed by a service chain $s_f \in S$ consists of a sequence of ordered VNFs, denoted as $s_f = \{m_j | j \in \{1, 2, \ldots, |s_f|\}\}$. When all VNF instances of a service chain s_f are active, s_f is called available. If no failure happens, all the active VNF instances of a service chain are called primary. When failure happens, the VNF instances that will be activated are called backups. The requested service chain availability is ρ_s and the actual reliability is $a(s)$. p_{ij} is the availability of VNF m_j on server v_i. ϕ_{ij} is the number of VNF m_j on server v_i. Another indicator parameter Y_{ij} is defined to demonstrate whether VNF m_j on v_i is available. We define an indicator parameter X_{ij} to show whether the backup instance of VNF f_i is deployed at the node v_j. If yes, $X_{ij} = 1$; otherwise, $X_{ij} = 0$. We define the set of VNFs that have their backups at a node v_j as D_j. We refer to the sequence $D = (D_1, D_2, \ldots, D_n)$ as a backup deployment of VNFs to nodes. For ease of reference, we list the notations in Tab. I. We formulate the availability of a service chain when both the hardware and software failures are possible as:

$$a = \sum_{\{X_l\}} \left\{ \prod^{i} q_i^{X_i} (1 - q_i)^{1 - X_i} \cdot \prod^{j} [1 - \prod^{i} (1 - p_{ij} \cdot X_i)^{\phi_{ij}}] \right\}$$

If the availability of each server is identical and the availability of a type of VNF m_j is irrelevant to its deployed server, then the availability of a service chain is

$$a = \sum_{\{X_l\}} \left\{ \prod^{i} q^{X_i} (1 - q)^{1 - X_i} \cdot \prod^{j} [1 - \prod^{i} (1 - p_j \cdot X_i)^{\phi_{ij}}] \right\}$$

If there is no hardware failure, the availability of a service chain can be simplified as

$$a = \prod^{j} [1 - \prod^{i} (1 - p_{ij})^{\phi_{ij}}]$$

B. Problem Formulation

In this paper, we aim at minimizing the total upper-bound transmission delay of all flows when both the hardware and software failures are possible. The deployment plan is denoted as $\Phi = \{\phi_{ij},\ \forall v_i \in V,\ m_j \in M\}$.

$$\min_{\Phi,P} \sum f\, d_f \tag{1}$$

$$\text{s.t. } d_f = \max(\sum e_{ij} \in p_f\, d_{ij}) \qquad \forall f \in F \tag{2}$$

$$\sum i\, phi_{ij} \leq c_j \qquad \forall v_i \in V, \forall m_j \in M \tag{3}$$

$$\sum e_{ij} \in p_f\, r_f \leq b_{ij} \qquad \forall e_{ij} \in E, \forall f \in F \tag{4}$$

$$a(s) \geq \rho_s \qquad \forall s \in S \tag{5}$$

$$X_i = \{0,1\}, Y_{ij} = \{0,1\} \qquad \forall v_i \in V, \forall m_i \in M \tag{6}$$

We aim at minimizing the total upper-bound latency of a function-backup allocation and assignment. We can formulate our optimization problem in a mathematical way as follows. 1 shows the objective of the optimization problem. The constraint in 2 illustrates d_f is the largest total transmission delay among all its possible routing path. In 3, we require that the total number of deployed backup instances on each node is within its capacity. 4 guarantees the link bandwidth capacity. We provide the availability guarantee in 5. 6 requires the indicator parameters X_i and Y_{ij} can only be assigned with a value of 0 or 1.

IV. Probabilistic Analysis

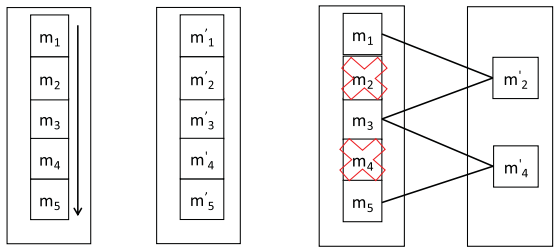
When we have the probabilistic prior failure information, we consider the problems of VNF deployment, resource allocation, and flow routing with resilience guarantees against both hardware and software failures. In certain scenarios, such information can be learned from failure analytic, using historical data. In this section, we first discuss the hardware failure and then the deployment of primary and backup instances.

A. Hardware Selection

First, we prove that if the hardware server can fail, then we should always deploy the active and the backup instances for one VNF at different servers.

Theorem 1: The active and backup instances for one VNF should be deployed in different servers in order to have a higher availability of the VNF.

Proof: We prove the theorem by probability analysis. We need there to be at least one instance between the active one and the backup one for each VNF ϕ that is available. If all n active instances and backup instances are deployed in the same server, the availability of the VNF ϕ is equal to $a_1 = q \cdot [1-(1-p)^n]$. If they are distributed to different servers, the availability of the VNF ϕ is equal to $a_2 = 1-(1-q\cdot p)^n$. Then we calculate the difference between these two values: $a_2 - a_1 = 1-(1-q\cdot p)^N - q\cdot[1-(1-p)^n]$. When $n=2$, we have $a_2 - a_1 = (2q\cdot p - (q\cdot p)^2) - (2q\cdot p - q\cdot p^n) = q\cdot p^2(1-q)$. We know that the availability of a server q is always less than 1, even though it is close to 1. Then we gave $a_2 - a_1 \geq 0$, illustrating that the distributed deployment has a higher availability. As a result, the active and backup instances need to be deployed in different servers. ■



(a) Without a failure (b) Upon failures.

Fig. 2. Illustration of service chain processing. (a) Without a failure. (b) Upon failures.

Then we give two specific failure probabilities of hardware to check the simultaneous server failure situation. When $p = 0.9$, the probabilistic is $(1-0.9)^2 = 10^{-2}$. When $p = 0.99$, the probabilistic is $(1-0.99)^2 = 10^{-4}$. Both values are extremely small, so we assume the simultaneous failures of two servers will not happen in this paper. Additionally, even if this happens, we can still use the resource in the cloud net [26]. As a result, we deploy the primary VNF instance and all its backup instances on only two servers in order to decrease the upper-bound transmission delay d_f.

B. Distribution of Primary and Backup VNF Instances

From above, we distribute primary and backup instances on only two servers. Here we use a specific example to illustrate the number distribution on two servers in order to maximize the availability of the VNF. For the same function with four instances including one primary and three backup instances, if we distribute them evenly on two servers, the availability is calculated as $a_1 = 1-(1-q)^2 - 2q\cdot(1-q)\cdot(1-p)^2 - q^2\cdot(1-p)^4$. If we distribute them one and three on two servers, the availability is calculated as $a_2 = 1-(1-q)^2 - q\cdot(1-q)\cdot(1-p) - q\cdot(1-q)\cdot(1-p)^3 - q^2\cdot(1-p)^4$. We find that when $p \geq 0.391$, $a_1 \geq a_2$. In real systems, p is definitely larger than 0.5. Therefore, we should distribute instances of the same function evenly on two servers.

Then, to satisfy the service chain processing requirement, we always deploy all types of VNF instances along flows' routing path according to the required order in the chain. We also need to make sure that each flow $f \in F$ meets its service chain availability requirement, which can be expressed as $a(s) \geq \rho_s,\ \forall f \in F$. Note that when the availability of the provided service chain can be satisfied, we should deploy as few backup instances as possible in order to save resources for future use. This problem has been proven to be NP-hard in [8]. It is out of scope of our paper, so we directly apply the proposed solutions in [8] to decide which functions should have backup instances.

For the upper-bound flow transmission delay d_f, we calculate the largest time among all its possible transmission path through all VNFs in its required service chain when we take all possible failure situations. We use Fig. 2 to illustrate the

Algorithm 1: Single Service Chain Solution with Homogeneous VNFs (SOV)

In: Flow f, its required service chain s_f and sets of vertices V, links E;
Out: The deployment plan and f's routing path;
1: Sort the server capacity $c_v, \forall v \in V$;
2: Apply Dijkstra's algorithm to find the k-shortest paths from src_f to dst_f until the total capacity of all servers in the path is no less than $|\phi_f|$;
3: Select the servers with the largest capacity to deploy the residual VNFs in sequence until fully deployed;
4: Return the deployment plan and f's routing path.

Algorithm 2: Single Service Chain Solution with Heterogeneous VNFs (SEV)

In: Flow f, its required service chain ϕ_f and sets of vertices V, links E;
Out: The deployment plan and f's routing path;
1: Sort the hop transmission delay of links.
2: Calculate the number of servers with a total capacity no less than $|\phi_f|$, which is $\lceil \frac{|\phi_f|}{c_v} \rceil$.
3: Use Alg. 1 to calculate the minimum total transmission delay from v_s to v_d with a path length as $\lceil \frac{|\phi_f|}{c_v} \rceil$.
4: Deploy the primary and backup VNF instances.
5: Return the deployment plan and f's routing path.

service chain processing. If there is no failure, shown in Fig. 2 (a), the flows get processed by all primary instances in one server along its path. However, if there are two failures of m_2 and m_4, shown in Fig. 2(b), the routing path look like a zig-zag, which includes extra link transmission delays among these two servers. We aim at minimizing the total upper-bound transmission delay of flows in this paper.

V. Single Service Chain Resilience

We start with a simple case in which there is only one service chain under two different configuration settings of VNFs: homogeneous or heterogeneous VNFs. In homogeneous VNFs, different types of VNFs consume the same amount of server resource while heterogeneous ones consume various amounts of server resources. Our problem becomes to deploy the requested service chain of f and find a path for f, in order to minimize its total transmission delay.

A. Single Service Chain With Homogeneous VNFs

We propose an optimal algorithm for the single service chain with homogeneous VNFs in Alg. 1. The insight of Alg. 1 is to select a path that consists of the servers with the largest capacity in order to shorten the flow routing path. Line 1 sorts the server capacity. Line 2 routes the flow f to the path with the total minimum transmission delay. Line 3 deploys the primary and backup instances of the service chain. The deployment plan and the routing path are returned in line 4. After we obtain the number of backup instances for each primary VNF in each service chain, we start to assign the instances to servers with the residual capacity and apply our Alg. 1. In Alg. 1, line 1 sorts the server capacity $c_v, \forall v \in V$. Line 2 applies Dijkstra shortest path algorithm to find the shortest path from src_f to dst_f. We select the servers with the largest capacity to deploy the residual VNFs in sequence until fully deployed in line 3 and return the deployment plan and its routing path in line 4.

Theorem 2: Our proposed Alg. 1 is optimal.

Proof: With homogeneous VNF instances, all VNFs consume the same amount of server resource and we can calculate the number of VNF instances that each server can support. As there is only one service chain, we can only keep all links with a sufficient bandwidth, meaning $b_e \geq r_f,\ \forall e \in E$. We have known the number of primary and backup VNF instances for each type in the service chain. Moreover, the sequence order restricts to deploy each type one by one along the flow path. In order to minimize the total transmission delay, Dijkstra's algorithm is optimal when finding a path with the minimized total cost. The cost is the latency of our resilient service chain model. We have already checked the deployment of all VNFs, which can guarantee the feasibility. We show the upper bound of the transmission delay does not affect the path with enough server capacity. Thus, our proposed SHV is optimal. ■

Time complexity: In Alg. 1, lines 1 and 3 take $O(|V|\log|V|)$ because of sorting. Line 2 runs the Dijkstra's algorithm, whose time complexity can be $O(n^2)$ [27], where n is the cardinality of one set. In our problem, since the deployment is feasible (the servers can support all the backup instances), we have $|V|$. Then the cardinality of one set in our transformed bipartite graph is $|V|$ and line 3 takes $O(|V|^2)$ time. As a result, the time complexity of Alg. 1 is $O(\max\{|V|\log|V|, |V|^2\})O(|V|^2)$.

B. Single Service Chain With Heterogeneous VNFs

For the case with heterogeneous VNFs, we also propose an optimal solution in Alg. 2. In Alg. 2, line 1 divides the resource at all nodes proportional to the traffic rate ratio of these two flows. We apply the corresponding solution for a single flow with its allocated server resource in line 2. Then we combine the residual node resource of f with the allocated resource for f in line 3, and similarly apply the solution for f in line 4. Line 5 returns the deployment plan. The time complexity is the same as Alg. 1. The extra challenge here is to generate an optimal solution or a performance-guaranteed solution with a better approximation ratio for two such flows without backups.

With heterogeneous VNFs, the deployment of the primary and backup instances becomes much more complicated. If there is no service chain, then our problem is similar to the traditional NP-hard problem of Knapsack. However, a service chain consists of multiple VNF instances in a sequence order, which limits the position selection of VNF instances. What's more, with an objective of minimizing the upper-bound of the transmission delay, the zig-zag routing path further restricts the possible positions of primary and backup VNF instances of the same type. Next we prove the optimality of our proposed solution, Alg. 2. We have known the number of primary and backup VNF instances for each type in the service chain.

Algorithm 3: Multiple Service Chains with Unlimited Server Capacity (MUC)

In: Sets of flow F, vertices V, links E;
Out: The deployment plan and routing paths;
1: Calculate the numbers of primary and backup of each VNF in each service chain;
2: Apply Multiple-commodity solution and get the routing paths;
3: **for** each flow $f \in F$ **do**
4: Deploy each VNF m along its routing path p_f;
5: Update the deployment plan for f;
6: **end for**
7: Return the deployment plan and routing paths.

Moreover, the sequence order is restricted to deploy each type one by one along the flow path.

Theorem 3: Our proposed Alg. 2 is optimal.

Proof: As we aim at minimizing the upper-bound of flow's transmission delay, the sequence order of a service chain restricts the position selection of the primary and backup instances. For Alg. 2, in order to minimize the total transmission delay, we need to find the minimum number of servers to hold all VNF instances. We have known the number of primary and backup VNF instances for each type in the service chain. Moreover, the sequence order restrict to deploy each type one by one along the flow path. If the server capacity is not enough to deploy the current VNF instance, we can directly deploy it on the next hop along its routing path. Then the delay will not change and the path is optimal with the minimum upper-bound transmission delay. ■

Time complexity: In Alg. 2, lines 1-4 and 6 spend at most $O(|V|\log|V|)$ because of sorting. Line 5 runs the Dijkstra's algorithm, whose time complexity can be $O(n^2)$, where n is the cardinality of one set. It applies the algorithm 1, whose time complexity is $O(|V|^2)$. In our problem, since the deployment is feasible (the servers can support all the backup instances), we have $|V|$. Then the cardinality of one set in our transformed bipartite graph is $|V|$ and line 5 spends $O(|V|^2)$ time. As a result, the time complexity of Alg. 2 is also $O(|V|^2)$.

VI. Multiple Service Chain Resilience

Multiple flows may compete for the same link bandwidth resource [28], which makes our problem much more challenging. In addition, various VNF primary and backup instances of different types can share a server's capacity. These constraints further complicate our problem, making it NP-hard.

A. Unlimited Server Capacity

First, we prove the NP-hardness of our problem.

Theorem 4: Our joint VNF deployment and flow routing problem is NP-hard with multiple flows even with unlimited server capacity.

Proof: When there is no constraint on the server capacity, the VNF instances can be deployed at any server along the routing path of the flow. Additionally, the VNF deployment has no effect on the upper-bound of flow transmission delay. Then there is no dependency relationship between the flow routing and VNF deployment. We can conduct these two parts, separately. The classic NP-hard multi-commodity problem [29] has the same objective as ours, which is minimizing the total transmission delay of all flows. It is reducible to our problem when there is even no need to deploy VNF instances. Therefore, the multi-commodity problem is a special case of our problem. The theorem holds that our problem is NP-hard. ■

As our problem is NP-hard with multiple flows even if there is no server capacity constraint, we propose a solution in Alg. 3, which utilizes the performance-guaranteed solution of the multi-commodity problem with an objective of minimizing the total edge cost [29]. In Alg. 3, line 1 divides the resource at all nodes proportional to the traffic rate ratio of these two flows. We apply the corresponding solution for a single flow with its allocated server resource in line 2. Then we combine the residual node resource of f with the allocated resource for f in line 3, and similarly apply the solution for f in line 4. Line 5 returns the deployment plan. The time complexity is the same as Alg. 1. Various VNF active and backup instances can share a server's capacity. These constraints further complicate our problem, making it challenging. When we have the probabilistic prior failure information of all hardware servers and software functions, we study the resource management problem for VNFs with the objective of minimizing the total transmission delay among all flows, whose service chain requests have been served by the system. First, we prove that if the hardware server can fail, then we should always deploy the active and the backup instances for one VNF at different servers. The extra challenge here is to generate an optimal solution or a performance-guaranteed solution with a better approximation ratio for the case. We leave the topic for our future work.

We illustrate the insight of this heuristic solution with the motivating example in Fig. 1. If we get the routing paths for all these three flows as shown in the figure, we need to deploy their requested primary and backup VNF instances along their paths. Because there is no server capacity, there is no competition on the server resources and the deployment for each flow is independent. Then we can simply deploy all VNF instances along their routing path by distributing all primary ones in a server and all backup ones in its next-hop sever.

Theorem 5: Our proposed Alg. 3 is performance-guaranteed with an approximation ratio as $\epsilon^{-2} \times |V|^2 \log b_{max}$, where ϵ is an arbitrary number and $b_{max} = \max_{e \in E} b_e$.

Proof: For Alg. 3, we apply the solution for the multi-commodity solution in [29], whose approximation ratio is $\epsilon^{-2} \times |V|^2 \log b_{max}$, where ϵ is an arbitrary number and $b_{max} = \max_{e \in E} b_e$. Our solution does not change the approximation ratio proof. Additionally, the deployment of primary and backup VNF instances for each flow is independent. The deployment has no influence on the objective. Therefore, Alg. 3 is performance-guaranteed with an approximation ratio as $\epsilon^{-2} \times |V|^2 \log b_{max}$. ■

Time complexity: In Alg. 3, lines 1 and 4 spend a constant time. The multi-commodity solution has a time complexity as $O(\epsilon^{-1} \times |F| \times |V|^2 \times \log b_{max})$ in line 2, where ϵ is an

Algorithm 4: Multiple Service Chains with Limited Server Capacity (MLC)

In: Sets of flow F, vertices V, links E;
Out: The deployment plan and routing paths;
1: Calculate the numbers of primary and backup of each VNF in each service chain;
2: **for** each flow $f \in F$ **do**
3: Apply Alg. 1 to achieve p_f and ϕ_f;
4: Update the remaining bandwidth in each edge $e \in p_f$ as $b_e = b_e - r_f$;
5: Update the remaining server capacity in each server c_v along p_f;
6: **end for**
7: Return the deployment plan and routing paths.

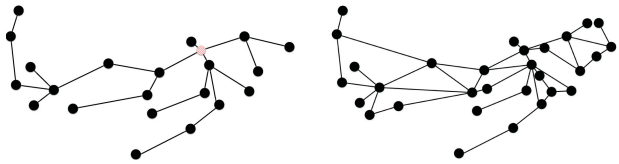
arbitrary number and $b_{max} = \max_{e \in E} b_e$. Lines 3-6 have $|F|$ loops and each loop takes at most $|V|$. Therefore, the time complexity of Alg. 3 is $O(\max\{|F| \times |V|, \epsilon^{-1} \times |F| \times |V|^2 \log b_{max}\}) = O(\epsilon^{-1} \times |F| \times |V|^2 \log b_{max})$, where ϵ is an arbitrary number.

B. Limited Server Capacity

When the server capacity c_v is limited, our problem is complicated. We introduce a heuristic solution for the limited server capacity case in Alg. 4. Every time, we randomly select to deploy the VNFs for each flow. Line 1 calculates all link transmission delays in an increasing order. Line 2 calculates the number of servers to deploy all the active instances. We apply the dynamic programming method to find the path with the minimum transmission delay in line 3. Line 4 sorts the availability of all backup instances in the service chain. Line 5 decides the functions with backup instances and deploys the backup instances primarily. For the other instances, we select the server to deploy them in line 6. Line 7 returns the final deployment plan. The solution is heuristic with the insight from our proposed solution in Alg. 1. Line 1 divides the resource at all nodes proportional to the traffic rate ratio of these two flows. We apply the corresponding solution for a single flow with its allocated server resource in line 2. Then we combine the residual node resource of f with the allocated resource for f in line 3, and similarly apply the solution for f in line 4. Line 5 returns the deployment plan. The time complexity is the same as Alg. 1. We illustrate the insight of this heuristic solution in Fig. 1. We note that the solution is heuristic. A performance-guaranteed solution will be sought if one exists will be part of future work.

We illustrate the insight of this heuristic solution with the motivating example in Fig. 1. If we get the routing paths for all these three flows as shown in the figure, we need to deploy their requested primary and backup VNF instances along their paths. Because of the limited server capacity, there is competition on the server resources and the deployment for each flow is dependent. Then we solves the problem by arranging flows one by one. We first select the flows with the maximum flow rate and apply our solution for a single service chain until all flows have been handled.

Time complexity: In Alg. 4, we have $|F|$ iterations in lines 2-6. For each iteration, we apply Alg. 1 in line 3, whose time complexity is $O(|V|^2)$. Therefore, the time complexity of Alg. 4 is $O(|F| \times |V|^2)$.



(a) Topology for single flow. (b) General topologies.

Fig. 3. Simulation topologies. (a) Topology for single flow. (b) General topologies.

VII. Simulations

Simulations are conducted and numerical evaluations are presented to demonstrate the efficiency of our solutions. After we present the network and flow settings, the results are shown from different perspectives to provide insightful conclusions.

A. Experimental Settings

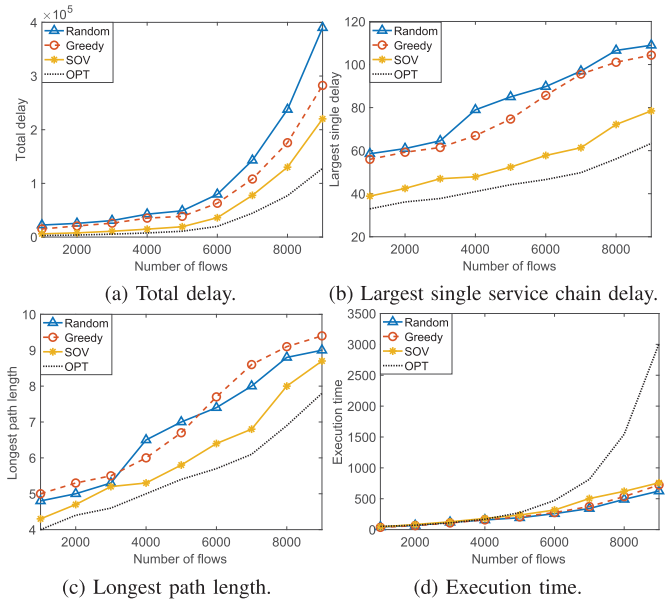
Topology: We conduct simulations by MATLAB on the Archipela-go (Ark) Infrastructure topology [30], which is CAIDA's active measurement infrastructure over United States serving the network research community since 2007. Additionally, traditional data center networks and WAN design over-provision the network with $30-40\%$ average network utilization in order to handle traffic demand changes and failures [31]. Thus, we assume each link has enough bandwidth to hold all flows. This assumption eliminates link congestion and ensures that the transmission of all flows is successful, since routing failure is none of our concern. We calculate the edge cost for the matching as the increment of the physical distance.

VNFs: There are 20 types of VNFs. The availability of active and backup instances for each type is a random number ranging from 0.7 to 0.9, respectively.

Service chain: Each service chain has a length ranging from 3 to 7, all of whose VNFs are selected from the 20 types. The availability requirement of each service chain is a random number ranging from 0.6 to 0.8. The variable is the number of incoming service chain ranging from 1000 to 9000 with a stride of 1000.

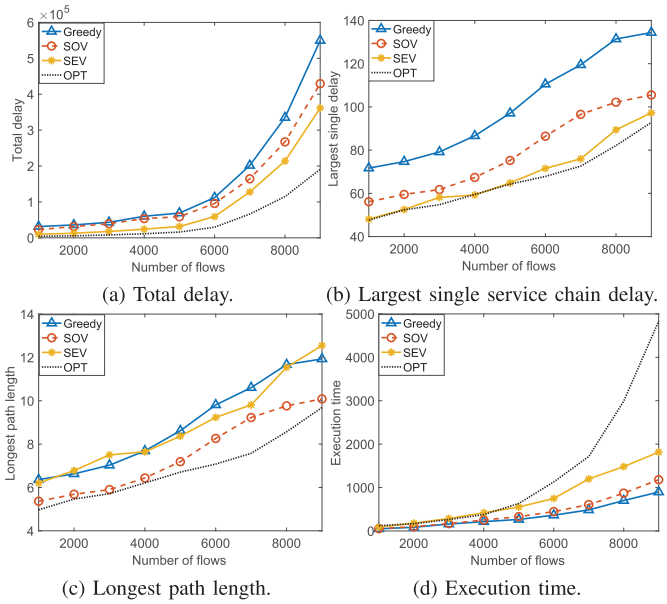
B. Comparison Algorithms and Performance Metrics

We conduct simulations for all above four cases, respectively. We include three comparison algorithms to evaluate each of our proposed solutions from different perspectives in each setting. The first algorithm for the case of single service chain with homogeneous VNFs is called Random, which randomly deploys the backup instances to any available server. The second algorithm for the two cases of single service chains is called Greedy, which greedily allocates each function of each service chain to its corresponding location with the minimum transmission delay increment. The third algorithm for all four cases is called OPT, which is the optimal solution obtained from running the Integer Programming



(a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.

Fig. 4. Single service chain with homogeneous VNFs. (a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.



(a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.

Fig. 5. Single service chain with heterogeneous VNFs. (a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.

Solver, Cplex [32]. For the single service chain with homogeneous VNFs, our solution is called SOV. For the heterogeneous VNF case, our solution is called SEV and we also include SOV as an comparison algorithm. For multiple service chains, our solution is called MUC for unlimited server capacity and MLC for limited server capacity. We use SOV and SEV as comparison algorithms for both cases. For each parameter setting, we run each algorithm multiple times and use the average as our final result into the simulations.

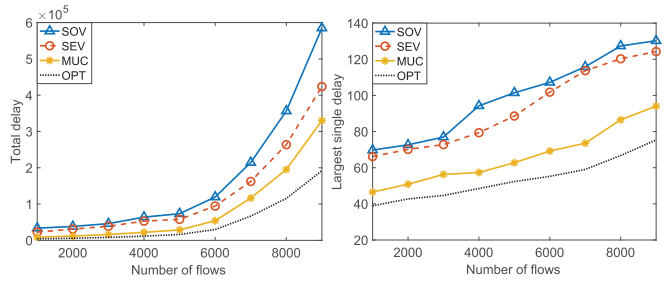
We use four performance metrics for our benchmark comparisons: the total upper-bound transmission delay, the maximum single service chain transmission delay, the maximum length of flows' routing path, and the execution time of running the algorithms. The total transmission delay is our objective in 1, which is the most important evaluation indicator. The maximum single service chain transmission delay is the largest total transmission delay among all flows. the maximum length of flows' routing path is the longest routing path length among all flows. The execution time is the time to obtain results by running the algorithms. The units of the total delay, the largest single service chain delay and the execution time are millisecond, millisecond and second, respectively.

C. Results for Single Service Chain With Homogeneous VNFs

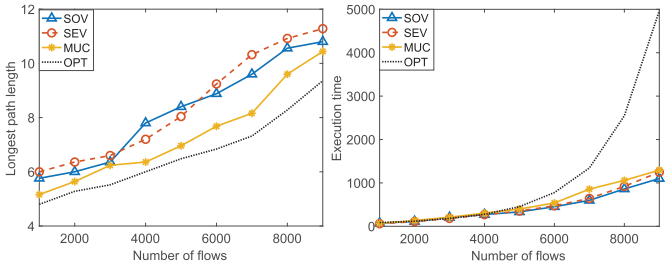
Fig. 4 shows the results of changing the number of flows from 1000 to 9000 when we have only one flow. Our proposed algorithm for this case is called Alg. SOV. Fig. 4(a) shows the result of the total delay. OPT has the best performance with the minimum total transmission delay, while our proposed SOV has the second smallest total delay. The delay of Alg. SOV is at most 23.1% more than that of the optimal solution. Alg. Random has the largest total delay because randomly deploying all VNF instances reduces the chance of finding a path with a smaller length. In Fig. 4(b), it shows the largest delay of a single flow when we change the number of flows. The largest delay among all flows becomes larger. When the number of flows increases, the largest single delays and the length of the longest path for running all algorithms become larger. This is because it is more difficult to find a shorter path for a flow to place all instances for its required service chain for a shorter path. The result for the longest flows' path is shown in Fig. 4(c). The path length becomes larger, especially for the Algs. Random and Greedy. The increment is more than 80.3% when the number of flows changes from 1000 to 9000. Our proposed SOV algorithm has a close performance with the optimal solution, which indicates its efficiency. Alg. Random has the worst performance with the largest path length. Fig. 4(d) shows the execution time of running all the algorithms. OPT needs to find the minimum delay value in each time, which makes it much more time-consuming than the others. The changing tendency of the execution time is in a form of exponential increment. The execution times of the other three algorithms are quite close, while the performance of our proposed Alg. SOV is much better than Algs. Random and Greedy.

D. Results for Single Service Chain With Heterogeneous VNFs

Fig. 5 shows the results of running all algorithms when we have one flow with heterogeneous VNFs. Our proposed algorithm for this case is called Alg. SEV. More VNF instances are deployed in order to meet the availability requirement of service chains. So the delay and the path length of each flow become a little bit larger. Fig. 5(a) shows the result of the total

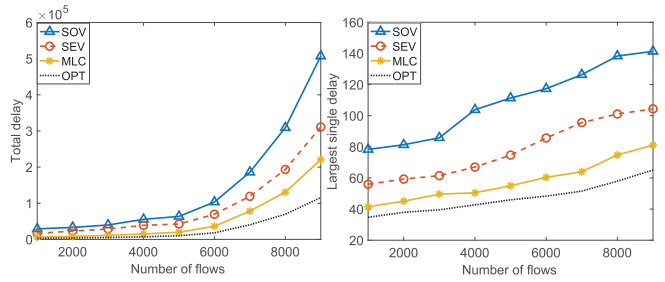


(a) Total delay. (b) Largest single service chain delay.

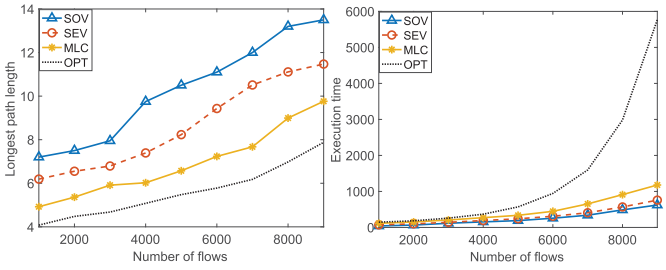


(c) Longest path length. (d) Execution time.

Fig. 6. Multiple service chains with unlimited server capacity. (a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.



(a) Total delay. (b) Largest single service chain delay.



(c) Longest path length. (d) Execution time.

Fig. 7. Multiple service chains with limited server capacity. (a) Total delay. (b) Largest single service chain delay. (c) Longest path length. (d) Execution time.

delay. OPT has the best performance with the minimum total delay while our proposed SEV has the second smallest delay. Alg. Greedy has the worst performance with the largest total transmission delay, which can be more than 5 times than that of Alg. OPT. This indicates that it is not enough to only deploy all backup instances on a single server, which directly results in a larger path length. In Fig. 5(b), the largest delay among all flows becomes larger because more flows are inserted, and less server capacity is left. When the number of flows increases, the increment of the total delay of all algorithms becomes larger. This is because it is more difficult to find a shorter path for a flow to place all instances for its required service chain. The path length of the newly-coming flow becomes longer, which has a direct impact on the total transmission delay. Fig. 5(c) shows the result of the largest path length of a single flow. Our proposed SEV algorithm has a close performance compared with the optimal solution, which indicates its efficiency. Fig. 5(d) shows the execution time of running all the algorithms. OPT needs to find the minimum delay value in each time, which makes it much more time-consuming than the other algorithms. The result demonstrates the trade-off between the performance and the efficiency of the algorithms. Our proposed Alg. SEV has a much lower execution time than Alg. OPT while their performances on the last three metrics are close.

E. *Results for Multiple Chains With Unlimited Server Capacity*

Fig. 6 shows the results of changing the number of flows from 1000 to 9000 when we have two flows without backup instances each time. Our proposed algorithm for this case is called Alg. MUC. Fig. 6(a) shows the result of the total delay. When the number of flows increases, the increment of the total delay of all algorithms becomes larger. This is because it is more difficult to find a shorter path for a flow to place all instances for its required service chain. The path length of the newly-coming flow becomes longer, which has a direct impact on the total transmission delay. OPT has the best performance with the minimum total delay while our proposed Alg. SEV has the second smallest total delay. In Fig. 6(b), the largest delay among all flows becomes larger because more flows are inserted, and less server capacity is left. Our proposed MUC algorithm has a close performance with the optimal solution, which indicates its efficiency. On average, the largest delay of Alg. MUC is just 27.9% more than that of the optimal solution. As for the result for the longest flows' path, shown in Fig. 6(c), Alg. SOV has the worst performance with the largest path length. This indicates that it is not enough to only deploy all backup instances on a single server, which directly results in a larger path length. Fig. 6(d) shows the execution time of running all the algorithms. OPT needs to find the minimum delay value in each time, which makes it much more time-consuming than the other algorithms. The execution time for this case is a little more than that of the last case because the placement of backup instances needs more time to decide beyond the active instances.

F. *Results for Multiple Chains With Limited Server Capacity*

Fig. 7 shows the results of changing the number of flows from 1000 to 9000 when we have two flows without backup instances each time. Our proposed algorithm for this case is called Alg. MLC. Fig. 7(a) shows the result of the total delay. When the number of flows increases, the increment of the total delay of all algorithms becomes larger. This is

because it is more difficult to find a shorter path for a flow to place all instances for its required service chain. The path length of the newly-coming flow becomes longer, which has a direct impact on the total transmission delay. OPT has the best performance with the minimum total delay while our proposed MLC has the second smallest total delay. In Fig. 7(b), the largest delay among all flows becomes larger because more flows are inserted, and less server capacity is left. Our proposed MLC algorithm has a close performance with the optimal solution, which indicates its efficiency. On average, the largest delay of Alg. MLC is just 33.7% more than that of the optimal solution. As for the result for the longest flows' path, shown in Fig. 7 (c), Alg. SOV has the worst performance with the largest path length. This indicates that it is not enough to only deploy all backup instances on a single server, which directly results in a larger path length. Fig. 7(d) shows the execution time of running all the algorithms. The execution time for this case is a little more than that of the last case because the placement of backup instances needs more time to decide where to deploy all primary and backup VNFs.

In summary, compared with several baselines, our proposed three algorithms always have excellent performances on all metrics, especially their designed objectives. The simulated results illustrate the importance of not only the link transmission delay, but also the server capacity. More VNF instances are deployed in order to meet the availability requirement of service chains, so the delay and the path length of each flow becomes a little bit larger. Additionally, we demonstrate the trade-off between the performance and the time-efficiency of our proposed algorithms.

VIII. Conclusion

One important issue of network services is reliability, which means that each type of VNF in a service chain acts properly on its function. The effective way to guarantee the reliability is to utilize sufficient redundancy through providing VNF backup instances beyond active primary ones. Software Defined Networking (SDN) enables the dedicate location pickup of primary and backup instances on switch-connected servers. The deployment of primary and backup VNF instances plays an important role of flow routing because of its influence on the transmission latency of the flow. In order to minimize the total latency of flows, we first formulate the reliable service chain deployment as a mathematical optimization problem. A detailed analysis of both software and hardware failure models is studied. Then we propose solutions for our problem. We start with the case of a single flow by introducing two optimal solutions for the homogeneous and heterogeneous VNFs with the same or different configurations. Next, we prove the NP-hardness of our problem for the case with multiple flows and propose an intuitive strategy. Additionally, a performance-guaranteed solution is included with an approximation ratio if the server capacity is infinite. Extensive simulations are conducted to evaluate our proposed solutions compared to multiple algorithms.

References

[1] Y. Chen and J. Wu, "Max Progressive Network Update," in *Proc. Int. Cartogr. Conf.* 2017.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem: Network Processing As a Cloud Service," in *Proc. SIGCOMM* 2012.

[3] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. Netw. Syst. Des. Implementation*, 2014.

[4] A. Gember-Jacobson *et al.*, "OpenNF: Enabling innovation in network function control," in *Proc. SIGCOMM* 2014.

[5] C. Voudouris, G. Owusu, R. Dorne, and D. Lesaint, *Service Chain Management: Technology Innovation for the Service Business*. Springer Science & Business Media, 2007.

[6] J. Sherry, S. Ratnasamy, and J. S. At, "A survey of enterprise middlebox deployments," Univ. California, Berkeley, Tech. Rep. UCB/EECS-2012-24, 2012. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-24.html

[7] Y. Chen, J. Wu, and B. Ji, "Virtual network function deployment in tree-structured networks," in *Proc. ICNP* 2018.

[8] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. INFOCOM* 2017.

[9] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A. field study of middlebox failures in datacenters," in *Proc. IMC* 2013.

[10] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 350–361, 2011.

[11] J. Wu, "Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 4, pp. 321–334, Apr. 1998.

[12] R. Xia, H. Dai, J. Zheng, R. Gu, X. Wang, and G. Chen, "Safe: Service availability via failure elimination through VNF scaling," in *Int. Conf. Proc. Ser.* 2019.

[13] S. Lal, T. Taleb, and A. Dutta, "NFV: Security threats and best practices," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 211–217, Aug. 2017.

[14] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tut.*, vol. 21, no. 2, pp. 1409–1434, Jul.-Aug. 2019.

[15] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *Proc. INFOCOM* 2018.

[16] 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. Surv. Tut.*, vol. 18, no. 1, pp. 236–262, Jan.–Mar. 2015.

[17] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. INFOCOM* 2017.

[18] T. Lukovszki, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *J. Parallel Distrib. Comput.*, 2018.

[19] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing SDN upgrades in ISP networks," in *Proc. INFOCOM* 2017.

[20] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," Ph.D. dissertation, Univ. Côte d'Azur, CNRS, Inria Sophia Antipolis, France, 2018.

[21] H. Yu, J. Yang, C. Fung, R. Boutaba, and Y. Zhuang, "ENSC: Multi-resource hybrid scaling for elastic network service chain in clouds," in *Proc. ICPADS* 2018.

[22] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," *Theor. Comput. Sci.*, vol. 496, pp. 184–194, 2013.

[23] T. Wang, H. Xu, and F. Liu, "Multi-resource load balancing for virtual network functions," in *Proc. ICDCS* 2017.

[24] S. G. Kulkarni, G. Liu, K. Ramakrishnan, M. Arumaithurai, T. Wood, and X. Fu, "Reinforce: Achieving efficient failure resiliency for network function virtualization based services," in *Proc. CoNEXT*, 2018.

[25] J. Liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, "Incremental server deployment for scalable NFV-enabled networks," in *Proc. INFOCOM* 2020.

[26] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," in *Proc. INFOCOM* 2020.

[27] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *J. ACM, (JACM)*, vol. 20, no. 3, pp. 385–388, 1973.

[28] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.
[29] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM Trans. Alg.*, vol. 4, no. 1, pp. 1–17, 2008.
[30] CAIDA. (2018) Archipelago Monitor Locations. [Online]. Available: http://www.caida.org/projects/ark/locations/
[31] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.,* vol. 43, no. 4, pp. 3–14, Aug. 2013.
[32] IBM, "IBM ILOG CPLEX V12. 1: Users manual for CPLEX," *IBM Corp.*, 2009, [Online]. Available: ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf

Yang Chen received the B.Eng. degree in electronic engineering and information science from the University of Science and Technology of China in 2015. She is currently the Ph.D. candidate with the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, USA. Her current research focuses on software defined networks, especially resource allocation, flow scheduling, and network updates.

Jie Wu (Fellow, IEEE) is the Director of the Center for Networked Computing and Laura H. Carnell Professor with Temple University. He also is the Director of International Affairs with the College of Science and Technology. He was the Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a Program Director with the National Science Foundation and was a Distinguished Professor with Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He is on several Editorial Boards, including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICE COMPUTING, *Journal of Parallel and Distributed Computing*, and *Journal of Computer Science and Technology*. He was the General Co-Chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program Co-Chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and Chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a Fellow of the AAAS. He was the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.