Availability- and Traffic-Aware Placement of Parallelized SFC in Data Center Networks

Meng Wang, Bo Cheng, *Member, IEEE*, Shangguang Wang, *Senior Member, IEEE*, and Junliang Chen

***Abstract*—Network Function Virtualization (NFV) brings flexible provisioning and great convenience for enterprises outsource their network functions to the Data Center Networks (DCNs). Network service in NFV is deployed as a Service Function Chain (SFC), which includes an ordered set of Virtual Network Functions (VNFs). However, in one SFC, the SFC delay increases linearly as the length of SFC increases. SFC parallelism can achieve high performance of SFC. In this article, we focus on the parallelized SFC placement problem in DCN considering availability guarantee and resource optimization. Firstly, we define the parallelized SFC and propose a multi-flow backup model. The parallelized SFC consists of multiple parallelized sub-SFCs. We split large data flow into multiple small sub-flows, each of them can be transmitted in one sub-SFC. The backup model provides backup sub-SFCs for working sub-SFCs to improve availability. Finally, we design three placement strategies and a Hybrid Placement Algorithm (HPA) aimed at mapping SFCs to DCN. Evaluation results show that our proposed solutions outperform the related work. We can reduce SFC delay (30%) and optimize link consumption (reduce 40%) while guaranteeing availability (99.999%).**

***Index Terms*—Network function virtualization, service function chain, placement, parallel, availability, traffic.**

I. Introduction

NETWORK Function Virtualization (NFV) [1] is a promising technology that decouples Network Functions (NFs) from the dedicated hardware. Due to the convenient and flexible management of Virtual Network Functions (VNFs), NFV significantly decreases the Capital Expenditure (CAPEX) and Operating Expense (OPEX) [2]. Nowadays, NFV is playing an important role in communication networks, mobile networks, enterprise networks, and Data Center Networks (DCNs). Based on multiple application requirements, network service is described as a Service Function Chain (SFC) [3], [4], which consists of an ordered set of VNFs.

Manuscript received April 27, 2020; revised October 22, 2020 and December 23, 2020; accepted January 4, 2021. Date of publication January 18, 2021; date of current version March 11, 2021. This work is supported by the National Key Research and Development Program of China under grant 2018YFB1003804, in part by the National Natural Science Foundation of China under grant 61921003, 61972043, and in part by the BUPT Excellent Ph.D. Students Foundation under grant CX2019214. The associate editor coordinating the review of this article and approving it for publication was J. Hwang. *(Corresponding author: Bo Cheng.)*

The authors are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: mengwang@bupt.edu.cn; chengbo@bupt.edu.cn; sgwang@bupt.edu.cn; chjl@bupt.edu.cn).

Digital Object Identifier 10.1109/TNSM.2021.3051903

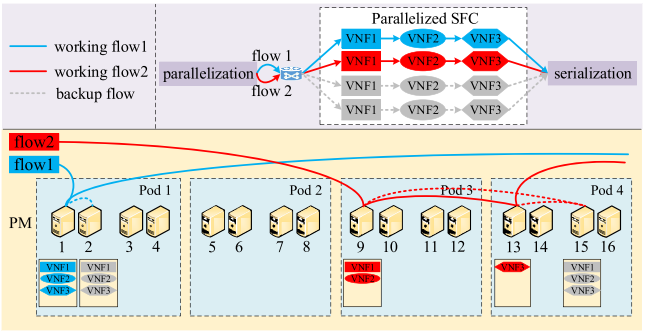


Fig. 1. Example of the parallelized SFC placement in DCN.

There are multiple applications with different requirements, such as delay, availability, and link consumption (i.e., the number of physical links used by SFC). For example, the SFC delay increases linearly as the length of SFC increases. For some delay-sensitive applications, this delay might be unacceptable. Availability is also an important issue. Compared with the traditional IT applications that require two 9s to three 9s. (i.e., 99% and 99.9%), the telecom applications often require four 9s or even five 9s [5]. For SFC placement, since the Physical Machines (PMs) in DCN are distributed, different placement strategies can result in different resource consumption (e.g., link and PM consumption). Therefore, SFC placement problem becomes an important but difficult problem, which has received increasing attention from both academic [6]–[8] and industry [9], [10].

However, there are still some problems remaining to be solved. Firstly, most of the existing work pay more attention to parallelized NF in the same SFC, not about the parallelized SFC. Secondly, most literature use traditional active/standby (i.e., $1 + 1$) redundancy models [11]–[13], in which a standby entity can be used if an VNF fails. However, these traditional backup models may not be suitable for guaranteeing the availability of parallelized SFC (elaborate in Section II). Thirdly, more existing work only focus on the availability guarantee in the VNF placement problem. However, increased availability leads to more link consumption.

As Fig. 1 shows, we illustrate the parallelized SFC placement problem by showing a simple example in a data center topology. There are four parallelized sub-SFCs, including two working sub-SFCs and two backup sub-SFCs. In this example, the large data flow is split into flow1 and flow2 (transmitted

by sub-SFC1 and sub-SFC2, respectively), which improves data transmission efficiency and reduces SFC delay. However, it is necessary to guarantee the availability of two working sub-SFCs to achieve high performance. Therefore, according to the redundancy mechanism, two backup sub-SFCs are provided to guarantee availability. Then, all sub-SFCs need to be mapped to DCN. As Fig. 1 shows, the link consumption of flow1 (blue solid line) and flow2 (red solid line) are different. Since the VNFs in flow1 are mapped in the same PM, there is no internal link consumption. VNFs in flow2 are deployed in a distributed manner with high link consumption. When a failure occurs, backup link consumption in flow2 (red dotted line) is also more than that in flow1 (blue dotted line).

The example above shows the parallelized SFC placement problem considering availability guarantee and link consumption optimization. Given these facts, we define the parallelized SFC consisting of multiple sub-SFCs. Our proposed parallelized SFC divides large data flow into multiple small sub-flows. To guarantee the availability of parallelized SFC, we propose a multi-flow backup model to provide backup sub-SFCs for working sub-SFCs. Then we design three placement strategies and a hybrid placement algorithm to map the parallelized SFC to DCN while reducing resource consumption.

In summary, the main contributions are as follows:

- Define the parallelized SFC and propose a multi-flow backup model. We divide large data flow into multiple small sub-flows, each of them can be transmitted in one sub-SFC. The backup model provides backup sub-SFCs for working sub-SFCs. We can improve data transmission efficiency, reduce SFC delay, and improve the availability of parallelized SFC.
- Design three placement strategies and a Hybrid Placement Algorithm (HPA). We divide the placement problem into three categories, each corresponds to a different affinity level. HPA maps VNFs to DCN according to placement strategy. By defining affinity domains for PM, link consumption can be effectively reduced.
- Evaluate the performance of our proposed backup model and placement algorithm. Compared with the related work, our proposed solutions can reduce SFC delay (30%) and optimize PL consumption (reduce 40%) while guaranteeing availability (99.999%).

The rest of this article is organized as follows. Section II discusses the related work. Section III describes the system model and formulates the parallelized SFC placement problem. Section IV discusses the proposed solutions including parallelized SFC, multi-flow backup model, placement strategy, and hybrid placement algorithm. Section V evaluates the performance of our proposed solutions. Finally, Section VI concludes this study.

II. Related Work

Recently, there are multiple solutions designed to solve the placement problem in NFV environment. In this article, we focus on the parallelized SFC placement problem considering availability guarantee and resource optimization. Therefore, we first discuss the existing placement algorithm considering resource optimization. Then we describe the related work including parallelism and availability.

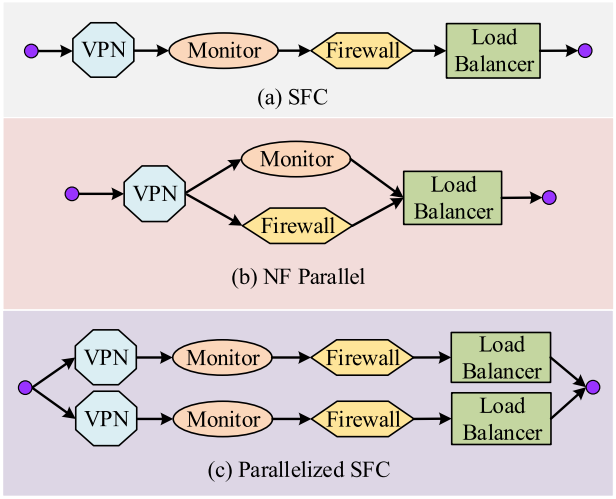


Fig. 2. Parallelism in NF and SFC.

A. Placement

Some existing work [14]–[18] solve the placement and scheduling problem considering different resource constraints and performance requirements. In [19], the authors propose a Local VNF Service Chain Placement (Local VSCP) Algorithm that keeps all VNFs as close as possible. Besides, some recent work solve the placement problem based on the NFV execution model. In summary, there are two existing NFV execution models: Run-To-Complete (RTC) [20]–[22] and Pipeline [23]–[25]. RTC aims to deploy SFC in a single PM, which can reduce PL cost. Pipeline deploys the same type of VNFs in a single PM to facilitate network operators to manage VNFs. In [26], the authors compare the performance of two NFV execution models. Tang *et al.* [27] design a dynamic VNF instance scaling system. And they propose two placement algorithms based on two NFV execution models in DCN.

Most solutions traditionally solve the placement problem (e.g., heuristic solutions). In this article, we design three placement strategies and a hybrid placement algorithm to combine the advantages of two existing NFV execution models.

B. Parallelism

Parallelism in packet processing brings significant delay optimization. Fig. 2(a) shows an SFC consisting of four NFs (i.e., VPN, Monitor, Firewall, and Load Balancer). Some excellent work such as NFP [28] and ParaBox [29] attempt to explore NF parallelism in NFV environment, as shown in Fig. 2(b). The principle of the existing work is to provide different packet copies for NFs running in parallel. For SFC parallelism, Engelmann and Jukan [30] propose the parallelized SFC by using Equal Cost Multipath Protocol (ECMP) [31], [32]. ECMP is aimed to balance the load when there are multiple flows of similar sizes.

By referring [30], we define the parallelized SFC including multiple sub-SFCs. Based on ECMP, we split the large data

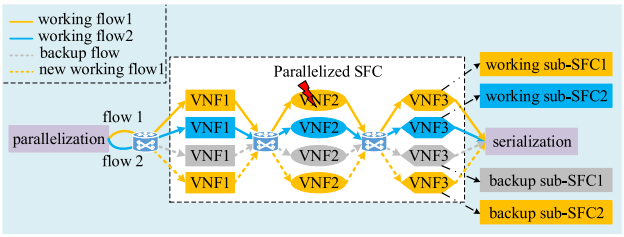


Fig. 3. Resource pool backup model.

flow into multiple parallelized small sub-flows. Each small flow is transmitted in one sub-SFC.

C. Availability

Most of the existing work guarantee availability by proposing backup models. Fan *et al.* [33]–[35] consider how to guarantee availability by using traditional active/backup model. Qu *et al.* [36], [37] research on the backup resource sharing when provisioning reliability-aware service chaining. However, the traditional backup models mostly provide one or more backup VNFs for one working VNF. Herker *et al.* [19] propose two backup models including VNF backup and SFC backup. For parallelized SFC backup model, Engelmann and Jukan [30] present a reliability study of parallelized SFC and propose a resource pool based backup model, as shown in Fig. 3. The main idea of resource pool backup model is to provide multiple backup VNFs for individual VNFs. Different from the existing approaches, we propose a multi-flow backup model, as shown in Fig. 8(a). The main idea of multi-flow backup model is to provide backup sub-SFCs for working sub-SFCs. We will describe it in detail in Section IV-B.

In summary, there are multiple solutions for parallelism, availability, and placement. However, most solutions cannot effectively solve the parallelized SFC placement problem. In this article, we define the parallelized SFC to enable SFC parallelism. And we propose a multi-flow backup model to guarantee the availability of parallelized SFC. Besides, we define three placement strategies and a hybrid placement algorithm to solve the placement problem as well as optimize resource consumption.

III. System Model and Problem Formulation

In this section, we describe the system model and problem formulation. First, we introduce the availability model. And then we propose the traffic model. Next, we describe two existing execution models of SFC. Finally, we formulate the SFC placement problem in DCN.

A. Availability Model

In this model, we define the availability of PM (and VNF), component, and SFC. In this article, one SFC is deployed as several components. And one component contains PM and the VNF which is running in this PM.

1) Availability of PM and VNF: In this model, the status of PM and VNF can be divided into uptime and downtime, which can be characterized in terms of Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR), respectively. Then the availability of PM and VNF can be characterized as follows:

$$A_{n_i}(A_{v_p}) = \frac{Uptime}{Uptime + Downtime} = \frac{MTBF}{MTBF + MTTR} \tag{1}$$

where A_{n_i} (or A_{v_p}) indicates the availability of PM n_i (or VNF v_p).

2) Availability of Component: Considering the availability of PM, we treat the PM and VNF running in it as a component c_e. We can conclude that the component is available when the PM and VNF are all available. Therefore, the availability of the component is:

$$A_{c_e} = A_{n_i} A_{v_p} \tag{2}$$

where A_{c_e} is the availability of component c_e. A_{n_i} is the availability of PM n_i, and A_{v_p} is the availability of VNF v_p which is running in n_i.

3) Availability of SFC: In this article, an SFC is deployed as an ordered set of components. As Fig. 4 shows, the components of SFC are usually organized in a sequential or parallel manner. We use A_{seq} or A_{para} to indicate the availability of the two kinds of components. A_{c_e} is the availability of component c_e, so is A_{c_f}. Then, the availability of the two components are as follows.

- *Sequential:* For the sequential manner, each component provides the different functions. Therefore, each component has to be available at the same time. The availability of the two components is:

$$A_{seq} = A_{c_e} A_{c_f} \tag{3}$$

- *Parallel:* We consider two parallel components that provide the same function. Therefore, the parallel way can be seen as the parallel of working path and backup path. At least one of the two paths is available to guarantee the overall availability. Therefore, the availability of the two components is:

$$\begin{aligned} A_{para} &= 1 - (1 - A_{c_e})(1 - A_{c_f}) \\ &= A_{c_e} + A_{c_f} - A_{c_e} A_{c_f} \end{aligned} \tag{4}$$

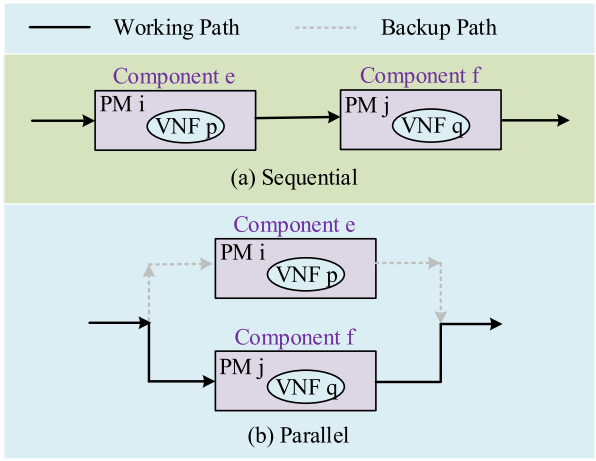


Fig. 4. Sequential and parallel.

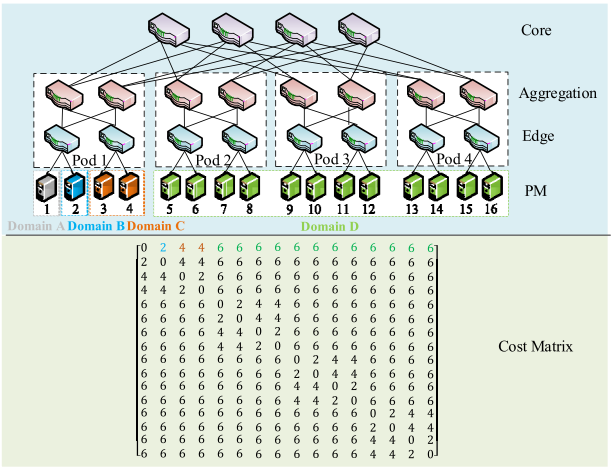


Fig. 5. Affinity domain in fat-tree topology.

In this article, we define that an SFC is deployed as a set of (sequential or parallel) components. Therefore, using the basic availability models (*Eqs.* (2), (3), and (4)) mentioned above, we can evaluate the availability of SFC:

$$A_{sfc} = \left(\prod_{c_e, c_f \in seq} A_{seq} \right) \left(\prod_{c_e, c_f \in para} A_{para} \right) \tag{5}$$

where A_{sfc} indicates the availability of SFC. However, the availability calculation of the parallelized SFC is more complex. We elaborate it in Section IV-C.

B. Traffic Model

There are four typical types of DCN typologies including Tree, VL2, Fat-tree, and BCube, which are described in detail in [38]. As Fig. 5 shows, we use a 3-layer fat-tree [39] data center topology as an example. The fat-tree contains three types of switches including core, aggregation, and edge switch. In a *h*-ray fat-tree, there are *h* pods, with *h*/2 aggregation switches and *h*/2 edge switches in each pod. Each pod is connected with $(h/2)^2$ core switches and with $(h/2)^2$ PMs.

As Fig. 5 shows, we can get the cost matrix based on the number of PLs between PMs. For example, the number of PLs between PM1 and PM2 is 2 (i,e., $M_{12} = 2$). The number of PLs between PMs can be divided into four types. Therefore, *M* is as follows:

$$M_{ij} = \begin{cases} 0 & \text{if } i = j \\ 2 & \text{if } \left\lfloor \frac{2(i-1)}{h} \right\rfloor = \left\lfloor \frac{2(j-1)}{h} \right\rfloor \\ 4 & \text{if } \left\lfloor \frac{2(i-1)}{h} \right\rfloor \neq \left\lfloor \frac{2(j-1)}{h} \right\rfloor \wedge \left\lfloor \frac{4(i-1)}{h^2} \right\rfloor \\ & \quad = \left\lfloor \frac{4(j-1)}{h^2} \right\rfloor \\ 6 & \text{otherwise } \left(\text{i.e., if } \left\lfloor \frac{4(i-1)}{h^2} \right\rfloor \neq \left\lfloor \frac{4(j-1)}{h^2} \right\rfloor \right) \end{cases} \tag{6}$$

where *i*, *j* are the IDs of PMs. The cost matrix *M* is a function of *h*, the total number of ports on each switch. For example, we compute M_{12} ($i = 1$ and $j = 2$). *h* is the number of ports on each switch ($h = 4$ in a 4-ray fat-tree). Therefore, we can conclude that $M_{12} = 2$ according to *Eq.* (6).

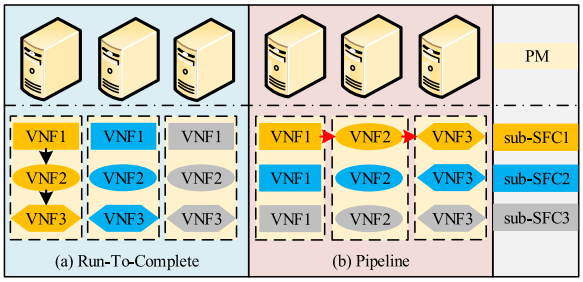


Fig. 6. NFV execution model: a) RTC and b) Pipeline.

In previous work [40], VNF is mapped based on affinity and anti-affinity constraints. In this article, we propose the affinity domain consisting of a set of PMs with the same link consumption. Based on these features, we divide the PMs into different affinity domains. As Fig. 5 shows, the affinity domain group of PM1 includes four affinity domains, which are A, B, C, and D, respectively.

C. Execution Model

As Figs. 6(a) and 6(b) show, we describe two existing NFV execution models: Run-To-Complete (RTC) [20]–[22] and Pipeline [23]–[25].

As Fig. 6(a) shows, RTC aims to deploy SFC in a single PM. In this model, we can ignore the Physical Link (PL) consumption (black line in Fig. 6(a)) of sub-SFC1 since there is no internal PL consumption between VNFs. However, the VNF is stateful when scaling in/out [41]. Therefore, we need to synchronize the VNF states, which requires more communication overhead.

As Fig. 6(b) shows, Pipeline deploys the same type of VNFs in a single PM. In this model, network operators can simplify the management of VNFs and easily discover which type of VNF is failing when troubleshooting. However, the PL consumption (red line in Fig. 6(b)) of sub-SFC1 in Pipeline cannot be ignored.

In summary, RTC and Pipeline have their pros and cons in terms of SFC performance and PL consumption. In this article, we are aimed at reducing PL consumption while guaranteeing availability in DCN. Therefore, we design three placement strategies (Section IV-C) to combine the advantages of RTC and Pipeline.

D. Problem Formulation

We use $PN^r = (N^r, L^r)$ to indicate the physical network. *t* indicates the size of physical network. For SFC, we use $SR^s = (V^s, L^s, A^s)$ to indicate the SFCR (SFC Request). $\mathbb{SR}$ indicates the set of SFCRs. Besides, *k* and *f* indicate the size of SFC and SFCR, respectively. A summary of used notation is found in Table I.

We aim to solve the parallelized SFC placement problem considering availability guarantee and PL consumption optimization. Different solutions (including backup models and placement algorithms) can result in different resource consumption. Therefore, we should meet the availability

TABLE I
BASIC NOTATIONS USED THROUGHOUT THIS ARTICLE

Symbol	Definition
Network	
$N^r = \{n_1^r, n_2^r \dots n_t^r\}$	the set of PMs. r is physical network.
$L^r = \{l_1^r, l_2^r \dots\}$	the set of PLs in r.
n_i^r, n_j^r	two PMs in r.
$l_{ij}^r = (n_i^r, n_j^r)$	the PLs between n_i^r and n_j^r.
SFC	
$\mathbb{SR}$	the set of SFCRs. $s \in \mathbb{SR}$ is an SFCR.
$V^s = \{v_1^s, v_2^s \dots v_k^s\}$	the set of VNFs in s.
$L^s = \{l_1^s, l_2^s \dots\}$	the set of logical links in s.
A^s	the availability requirement of SFC s.
v_p^s, v_q^s	two VNFs in s.
$l_{pq}^s = (v_p^s, v_q^s)$	the logical links between v_p^s and v_q^s.
Resource	
$cpu_{v_p^s}$	the CPU consumption of v_p^s.
$mem_{v_p^s}$	the memory consumption of v_p^s.
$bw_{l_{pq}^s}$	the bandwidth consumption between v_p^s and v_q^s.
$C_{n_i^r}^{cpu}$	the CPU capacity of n_i^r.
$C_{n_i^r}^{mem}$	the memory capacity of n_i^r.
$C_{l_{ij}^r}^{bw}$	the bandwidth capacity between n_i^r and n_j^r.
Variables	
$\alpha_{n_i^r}^{v_p^s}$	whether v_p^s is mapped in n_i^r.
$\beta_{l_{ij}^r}^{l_{pq}^s}$	whether logic link l_{pq}^s is mapped in PL l_{ij}^r.

constraints and placement constraints. In this article, we use PL consumption as our objectives.

1) Availability Guarantee: Firstly, we meet the availability requirements of SFCR. The availability of SFC is not less than the availability requirements. Therefore, the availability constraints can be formulated as:

$$A_{sfc} \geq A^s \tag{7}$$

where A_{sfc} indicates the availability of parallelized SFC, which we will elaborate in Section IV-C (*Eq.* (16)).

2) Placement and Resource Constraints: Then, we describe the constraints. We use $\alpha_{n_i^r}^{v_p^s}$ to describe whether VNF v_p^s is mapped in PM n_i^r:

$$\alpha_{n_i^r}^{v_p^s} = \begin{cases} 1 & \text{if } v_p^s \text{ is mapped in } n_i^r \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

We take CPU and memory constraints into consideration. In detail, the resource constraints of all VNFs placed in the same PM cannot exceed the resource capacity of this PM. So the constraints of CPU are:

$$\sum_{s \in \mathbb{SR}} \sum_{v_p \in V^s} cpu_{v_p^s} \cdot \alpha_{n_i^r}^{v_p^s} \leq C_{n_i^r}^{cpu} \tag{9}$$

Similarly, we formulate the memory constraints as:

$$\sum_{s \in \mathbb{SR}} \sum_{v_p \in V^s} mem_{v_p^s} \cdot \alpha_{n_i^r}^{v_p^s} \leq C_{n_i^r}^{mem} \tag{10}$$

Then we use $\beta_{l_{ij}^r}^{l_{pq}^s}$ to describe whether logical link l_{pq}^s is mapped in PL l_{ij}^r:

$$\beta_{l_{ij}^r}^{l_{pq}^s} = \begin{cases} 1 & \text{if } l_{pq}^s \text{ is mapped in } l_{ij}^r \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Similarly, the bandwidth consumption of the logical links placed on the same PL cannot exceed the capacity of this PL. Therefore, the bandwidth constraints are:

$$\sum_{s \in \mathbb{SR}} \sum_{l_{pq} \in L^s} bw_{l_{pq}^s} \cdot \beta_{l_{ij}^r}^{l_{pq}^s} \leq C_{l_{ij}^r}^{bw}. \tag{12}$$

3) Objectives: We focus on resource optimization including PM consumption and PL consumption.

As we discussed in the example in Fig. 1, different backup models and placement algorithms can result in different resource consumption, especially PL consumption. Therefore, we choose to optimize PL consumption, indicated as PL cost. It is worth mentioning that our solution to this optimization problem can also optimize PM consumption.

In order to get the PL cost, we calculate the number of PLs according to the logical link. As we mentioned, each logical link (e.g., l_{pq}^s) connects two VNFs (v_p^s and v_q^s). If one logical link is placed in one PL, we can get two PMs (n_i^r and n_j^r) running the two VNFs (i.e., v_p^s and v_q^s). The number of PLs between the two PMs is determined by cost matrix M that we mentioned in traffic model (Section III-C). Therefore, the total PL cost can be indicated as:

$$P = \sum_{s \in \mathbb{SR}} \sum_{l_{pq} \in L^s} \beta_{l_{ij}^r}^{l_{pq}^s} \cdot M_{ij} \tag{13}$$

Our goal is to minimize the PL cost:

$$\begin{aligned} &\min \ P \\ &s.t. \ \ Eq.\ 7 \ to \ Eq.\ 12. \end{aligned} \tag{14}$$

In summary, we formulate the parallelized SFC placement problem aiming at minimizing the PL cost.

IV. PROPOSED SOLUTION

In the proposed solution, firstly, we define the parallelized SFC. Secondly, we propose a multi-flow backup model. Thirdly, we define three placement strategies based on RTC and Pipeline. Finally, we design a hybrid placement algorithm.

A. Parallelized SFC

Traditionally, network service is deployed as an SFC. However, SFC delay increases linearly as the length of SFC increases. Therefore, we define the parallelized SFC by using the Equal Cost Multipath Protocol (ECMP) [31], [32]. In the parallelized SFC, the data in each sub-SFC becomes smaller. Therefore, the data is processed in parallel to improve data transmission efficiency and reduce SFC delay.

As Fig. 7(a) shows, there are three working sub-SFCs. The parallelization module splits large data flow into multiple sub-flows with a similar size. Switches can forward different sub-flows to the corresponding working sub-SFCs. There is one sub-flow in each sub-SFC being transmitted. Finally, the

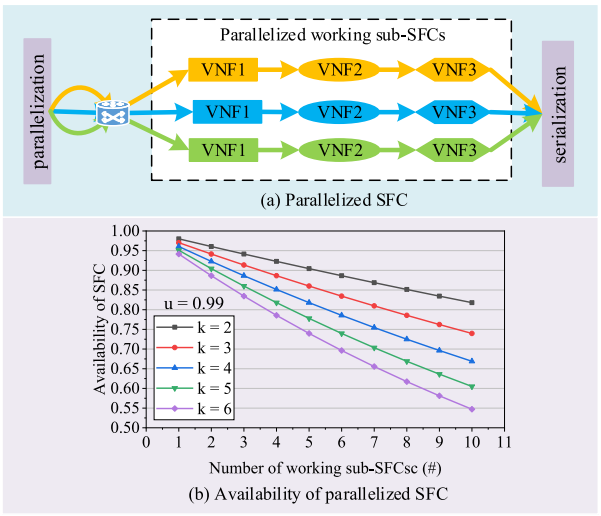


Fig. 7. (a) Parallelized SFC; (b) Availability of parallelized SFC.

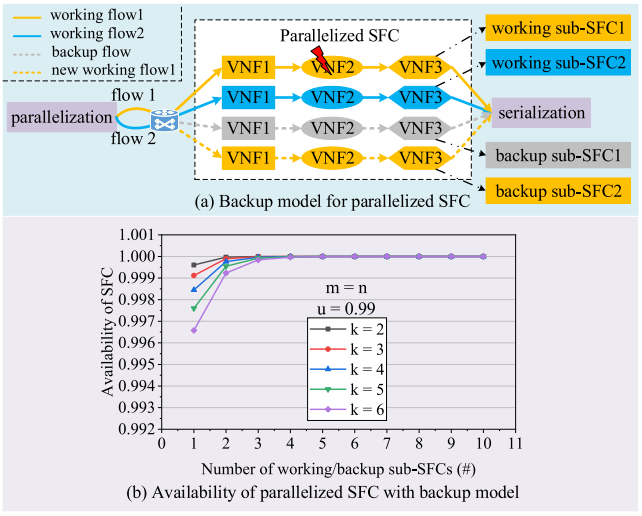


Fig. 8. (a) Multi-flow backup model; (b) Availability of parallelized SFC with backup model.

serialization module can merge data from each sub-SFC. In the parallelized SFC, some information and state between the same type of VNFs also need to be synchronized [41].

To achieve high performance and low SFC delay, VNFs in all working sub-SFCs need to be available. However, in this case, the overall SFC availability might be low. We use u to indicate the availability of VNF and assume $u = 0.99$. Therefore, the availability of parallelized SFC is:

$$A_{sfc} = \left(u^k\right)^m \tag{15}$$

Fig. 7(b) shows the availability of parallelized SFC under different numbers of VNFs and working sub-SFCs. We can observe that the highest availability is 98%. Moreover, availability decreases as the number of working sub-SFCs or VNFs increases. It is unacceptable for telecom applications.

Therefore, we next propose a backup model to improve the availability of parallelized SFC.

B. Backup Model

As Fig. 8(a) shows, we propose a multi-flow backup model to improve availability by providing backup sub-SFCs for working sub-SFCs. There are two working sub-SFCs and two backup sub-SFCs, as shown in Fig. 8(a). We can adjust the number and proportion of working sub-SFCs and backup SFCs based on resource constraints and availability requirements.

When a failure occurs in a working sub-SFC, the backup sub-SFC starts working. As Fig. 8(a) shows, we use an example to elaborate on the failure scenario. When VNF2 in working sub-SFC1 fails, the switch redirects flow1 to backup sub-SFC2. The yellow dotted line indicates the new working flow1. However, the resource pool backup [30] provides a resource pool for each individual VNF. In this model, more switches are needed. As Fig. 3 shows, when VNF2 in working sub-SFC1 fails, the switch redirects flow1 from VNF1 (in working sub-SFC1) to VNF2 (in backup sub-SFC2), which is different from our proposed multi-flow backup model.

Therefore, compared with the resource pool backup model, our proposed backup model has the following benefits:

- The multi-flow backup model does not need too many switches (which also have failures [42]) and can reduce the PL cost when failures occur while the resource pool backup model cannot.
- The multi-flow backup model can update the availability in an acceptable time. When some VNFs in a sub-SFC can not be placed in the same PM, the availability evaluation method of the resource pool backup model may become too complex to evaluate.

For availability, we assume that there are m working sub-SFCs, n backup sub-SFCs and each sub-SFC with k VNFs. The notations used can be founded in Table II. Fig. 8(b) shows the availability of parallelized SFC calculated according to *Eq.* (18) (just set $a = 1$ and $b = 1$ to ignore the availability of PM). We can observe that all availabilities of parallelized SFC with the backup model are higher than 99.6% when the numbers of working sub-SFC and backup sub-SFCs are the same. Moreover, the availability is close to 1 when the number of working sub-SFCs is 3 to 10. Compared with Fig. 7(b), we can conclude that the multi-flow backup model can effectively improve availability.

In the availability model, we take both VNF failures and PM failures into consideration. We show the availability calculation method that considers PM failures in Section IV-C.

In summary, the multi-flow backup model can effectively improve availability. However, the backup model increases resource consumption. In the trade-off between availability guarantee and resource consumption, we first meet the availability requirements. Once the availability requirements are met, we minimize resource consumption. For example, when we have a requirement of four 9s, we use one backup sub-SFC to reach three 9s, two can reach four 9s, and three can reach five 9s. In this case, we choose to use two sub-SFCs instead of three. This idea is also reflected in the placement algorithm.

C. Placement Strategy

After providing backup sub-SFCs for parallelized working sub-SFCs, we need to map all sub-SFCs to DCN. In this subsection, we present three placement strategies.

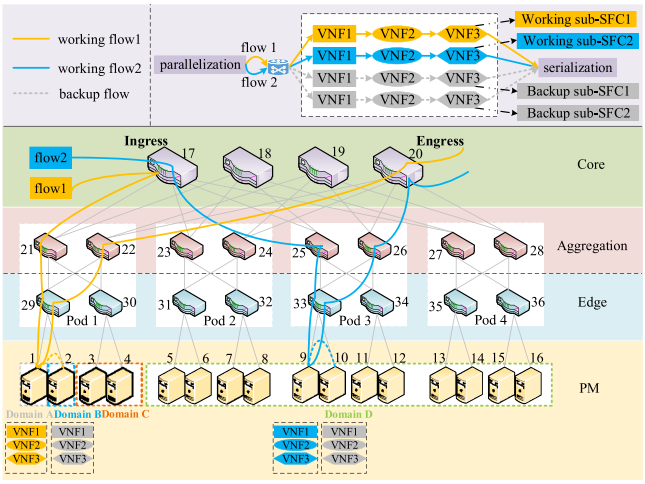


Fig. 9. Placement strategy of parallelized SFC.

Based on RTC and Pipeline (Section III-D), we divide the placement problem into three categories and define three corresponding placement strategies. The High-Affinity (S1) and Low-Affinity (S2) are designed to reduce PL cost between VNFs. Anti-Affinity (S3) is aimed at saving affinity domains.

- *S1 (High-Affinity): VNFs in the same sub-SFC.* Based on RTC execution model [22] and local placement [19], VNFs in working sub-SFC1 are in the same PM (i.e., Domain A in Fig. 9). There is no internal PL cost between VNFs. In this article, both working sub-SFCs and backup sub-SFCs are placed with High-Affinity.
- *S2 (Low-Affinity): Working sub-SFC and backup sub-SFC.* When a failure occurs, backup sub-SFC has to synchronize the status of VNFs from working sub-SFC, thus occurring extra PL cost. Low-Affinity placement (not be placed in the same PM) between working sub-SFC and backup sub-SFC can reduce PL cost when failures occur. This strategy is based on Pipeline model [23]. For example, PL cost between PM1 and PM2 can be reduced, as shown in Fig. 9.
- *S3 (Anti-Affinity): Working sub-SFC and working sub-SFC.* As Fig. 9 shows, there is no PL cost between working sub-SFCs. Affinity placement of two working sub-SFCs occupies affinity domains, which can be used to place backup sub-SFCs. Therefore, in order to save high affinity domains, we design this Strategy. For example, working sub-SFC2 is preferentially placed in Domain D (Anti-Affinity) of working sub-SFC1.

In summary, we map the parallelized SFC to DCN according to the above three placement strategies. To be closer to reality, we take into account both VNF failures and PM failures when evaluating the availability of parallelized SFC. The notations used are described in Table II.

In this article, we assume that θ PMs can hold one sub-SFC. We consider two available cases of the parallelized SFC: C_1 and C_2. In detail, C_1 indicates that all working sub-SFC are available. C_2 indicates that $i \in [1, min\{m, n\}]$ working

TABLE II
VALUES

Symbol	Value	Definition
a	[0.99, 0.9999]	working PM availability.
b	[0.99, 0.9999]	backup PM availability.
u	[0.9, 0.999]	working VNF availability.
v	[0.9, 0.999]	backup VNF availability.
m	3 (default)	number of working sub-SFCs.
n	3 (default)	number of backup sub-SFCs.
k	3 (default)	number of VNFs in each sub-SFC.
t	16 (default)	number of PMs in DCN.
A^s	one 9 $\sim$ five 9s	availability level.

sub-SFC fail. Therefore, $C_1 + C_2$ can cover all available cases of the parallelized SFC.

In C_1, all working sub-SFCs are available, which means that all working PMs and VNFs are available. Therefore, the availability of C_1 is:

$$A_{C_1} = \left(a^\theta u^k\right)^m \tag{16}$$

For C_2, we assume that there are $i \in [1, min\{m, n\}]$ working sub-SFCs failed. And we assume that there are $j \in [i, min\{m, n\}]$ backup sub-SFCs available to ensure the parallelized SFC available. Therefore, the availability of C_2 can be generalized as:

$$A_{C_2} = \sum_{i=1}^{min\{m,n\}} \binom{m}{i} \left(a^\theta u^k\right)^{m-i} \left(1 - a^\theta u^k\right)^i \times \sum_{j=i}^{min\{m,n\}} \binom{n}{j} \left(b^\theta v^k\right)^j \left(1 - b^\theta v^k\right)^{n-j} \tag{17}$$

Based on *Eqs.* (5), (16), and (17), the availability of parallelized SFC can be generalized as:

$$\begin{aligned} A_{sfc} &= A_{C_1} + A_{C_2} \\ &= \left(a^\theta u^k\right)^m + \sum_{i=1}^{min\{m,n\}} \binom{m}{i} \left(a^\theta u^k\right)^{m-i} \\ &\quad \times \left(1 - a^\theta u^k\right)^i \\ &\quad \times \sum_{j=i}^{min\{m,n\}} \binom{n}{j} \left(b^\theta v^k\right)^j \left(1 - b^\theta v^k\right)^{n-j}. \end{aligned} \tag{18}$$

D. Placement Algorithm

We design a Hybrid Placement Algorithm (HPA) that combines the advantages of RTC and Pipeline by using three placement strategies. Then we analyze the time complexity.

1) Framework of HPA: In HPA, we map the parallelized SFCs in DCN. For each SFC, we first map working sub-SFCs based on the placement strategy. Then we provide backup sub-SFCs for working sub-SFCs based on the multi-flow backup model. And the backup sub-SFCs are also mapped according to the placement strategy.

Algorithm 1: HPA (Hybrid Placement Algorithm)

Input: The physical network *r*: $PN^r = (N^r, L^r)$;
The set of SFCs: $\mathbb{SR}$;
Output: The placement result: PR_{res};
1 **Initialize:** The flag physical machine: PM_{flag};
2 **foreach** SFC^s *in* $\mathbb{SR}$ **do**
3 **foreach** $sub\text{-}SFC_i$ *in working sub-SFCs* **do**
4 $PM_{flag} = \textbf{PlaceOneSFC}(PM_{flag}, sub - SFC_i,$ Anti);
5 Update $PN^r = (N^r, L^r)$;
6 **end**
7 **foreach** $sub\text{-}SFC_i$ *in backup sub-SFCs* **do**
8 Get one PM that contains VNF as PM_{flag};
9 $PM_{flag} =$ ***PlaceOneSFC***$(PM_{flag},$ *sub-SFC*$_i$, Low);
10 Update $PN^r = (N^r, L^r)$;
11 Update A_{sfc} by using *Eq.* (18);
12 **if** $A_{sfc} \geq A^s$ **then**
13 Update PR_{res};
14 **end**
15 **end**
16 **end**
17 **return** PR_{res};

Function 1: ***PlaceOneSFC()*** Function

Input: The flag physical machine: PM_{flag}; One
sub-SFC: $SR^s = (N^s, L^s, A^s)$;
Affinity Level: *AL*;
Output: The flag physical machine: PM_{flag};
1 $DG_{pm} = \textbf{DomainGroup}(PM_{flag}, AL)$;
2 **foreach** VNF_p *in* SFC_i **do**
3 **foreach** PM_j *in* G_{pm} **do**
4 **if** PM_j *can hold* VNF_p **then**
5 Map VNF and add result into PR_{res};
6 $PM_{flag} = PM_j$;
7 break;
8 **end**
9 **else**
10 Continue to the next PM in G_{pm};
11 **end**
12 **end**
13 $DG_{pm} =$ ***DomainGroup***$(PM_{flag},$ *High*$)$;
14 **end**
15 **return** PM_{flag};

Function 2: ***DomainGroup()*** Function

Input: The physical machine: PM_i;
The Affinity level: *AL*;
Output: The affinity domain group result: DG_{pm};
1 **foreach** PM_j *in* PN^r **do**
2 According to *Eq.* (6), which one affinity domain (*A, B, C, D*) PM_j belongs to PM_i is caculated;
3 **end**
4 **if** *AL is High* **then**
5 Add *A, B, C* and *D* into DG_{pm} in order;
6 **end**
7 **if** *AL is Low* **then**
8 Add *B, C, D* and *A* into DG_{pm} in order;
9 **end**
10 **if** *AL is Anti* **then**
11 Add *D, C, B* and *A* into DG_{pm} in order;
12 **end**
13 **return** DG_{pm};

2) HPA: In general, we map a set of SFCs in this algorithm. For each SFC, working sub-SFCs are mapped first, and then backup sub-SFCs are mapped. For each sub-SFC, the algorithm calls *PlaceOneSFC* function.

The input is the physical network and the set of SFCs. And the output is placement result. First, we randomly choose one PM as PM_{flag}. It indicates the PM that be mapped currently. In this article, PM_{flag} is an important parameter to ensure the affinity between VNFs and affinity between sub-SFCs.

At lines 2-16 in Algorithm 1, we map a set of SFCs. Then lines 3-6 and 7-15 show the placement processes of working sub-SFCs and backup sub-SFCs, respectively. The two processes map each sub-SFC by calling Function 1 (lines 4 and 9 in Algorithm 1). Then at lines 5 and 10 in Algorithm 1, update the state of DCN. Unlike the working sub-SFC, the backup placement process updates the availability by using *Eq.* (18) after one sub-SFC is placed (line 11 in Algorithm 1). If the availability of SFC meets the requirements, the algorithm updates the placement result (lines 12-14 in Algorithm 1). Then, the algorithm starts the placement of the next SFC.

The *PlaceOneSFC* function is responsible for mapping one sub-SFC. In this function, each VNF in SFC is traversed until all VNFs are mapped. The mapping process of each VNF is based on the results of the *DomainGroup* function.

The input of *PlaceOneSFC* function is a flag physical machine, one sub-SFC, and the affinity level. In *PlaceOneSFC* function, we get the affinity domain group of PM_{flag} according to placement strategy (line 1 in Function 1). The affinity domains are used to control the affinity between sub-SFCs. For each VNF in sub-SFC, the function traverses all PMs in the affinity domain group until one PM can hold this VNF (lines 2-4 in Function 1). If the PM can hold this VNF, map VNF to PM and record the placement results (line 5 in Function 1). Otherwise, the algorithm checks if the next PM can hold this VNF (lines 9-11 in Function 1). At line 6 in Function 1, update this PM as the new PM_{flag}. After mapping one VNF, the function gets the affinity domains that are used to control the affinity (High-Affinity) of VNFs in the same sub-SFC (line 13 in Function 1). Finally, the algorithm returns a flag physical machine that is used to control the affinity of sub-SFCs (line 15 in Function 1).

The *DomainGroup* function is responsible for calculating the affinity domain group of a certain PM. According to *Eq.* (6), the function traverses each PM to determine which affinity domain it belongs to (lines 1-3 in Function 2). Then at lines 4-12 in Function 2, an affinity domain group is generated according to the affinity levels (e.g., High, Low, and Anti).

3) Complexity Analysis: As we mentioned, m, n, f, k and t indicate the number of working sub-SFCs, backup sub-SFCs, SFCs, VNFs in each sub-SFC, and PMs in DCN, respectively.

In Function 2, the *foreach* (line 1) runs t times. Therefore, the time complexity of Function 2 is at the level of $O(t)$.

In Function 1, the two *foreach* (lines 2 and 3) run k and t times, respectively. At lines 1 and 13, the function calls Function 2. Therefore, the time complexity of Function 1 is: $O(t + k(t + t))$. We can conclude that the time complexity of Function 1 is at the level of $O(kt)$.

In Algorithm 1, the *foreach* (line 2) runs f times. The two *foreach* (lines 3 and 7) run m and n times, respectively. At lines 4 and 9, the algorithm calls Function 1. Since the number of sub-SFCs can be determined by calculating the availability in advance, we do not consider this within the total time complexity. Therefore, the time complexity of Algorithm 1 is: $O(f(mkt + nkt))$.

In summary, we can conclude that the time complexity of the proposed Hybrid Placement Algorithm (HPA) is at the level of $O((m + n)fkt)$.

V. Performance Evaluation

A. Experiment Setup

We evaluate the performance using a machine of Ubuntu 18.04 with Intel CPU (2.10 GHz, 16 cores) and 256 GB memory. The soft switch we used is BESS (Berkeley Extensible Software Switch) [43], a modular framework integrated with DPDK [44]. And we use Alevin [6], [45], a widely used environment for NFV resource allocation. Based on BESS, we implement the parallelized SFC. Then we implement the backup model and placement algorithm based on Alevin.

By referring to Google Apps [46] and other literature [19], [34], we set the values of the parameters (Table II).

In this experiment, we use the parallelized SFC consisting of NAT (Network Address Translation), FW (Firewall), Proxy, and IDS (Intrusion Detection System). The total number of packets per second to generate is 10^6. The computing and memory requirements of each VNF are between [10, 20] and [5, 10]. The bandwidth requirements of each logical link are between [10, 30].

For data center networks, the computing, memory, and bandwidth capacities are all between [50, 100]. The number of PMs in DCN is among {16, 54, 128, 250}.

B. Comparison Backup Models and Placement Algorithms

1) Backup Models: We compare our proposed multi-flow backup model with resource pool backup model.

- Resource pool backup model [30] provides backup VNFs for each individual VNF in sub-SFC.
- Multi-flow backup model provides backup sub-SFCs for working sub-SFCs.

2) Placement Algorithms: We evaluate our proposed HPA with Local VSCP and resource pool based placement algorithm. Besides, we implement RPA as the baseline.

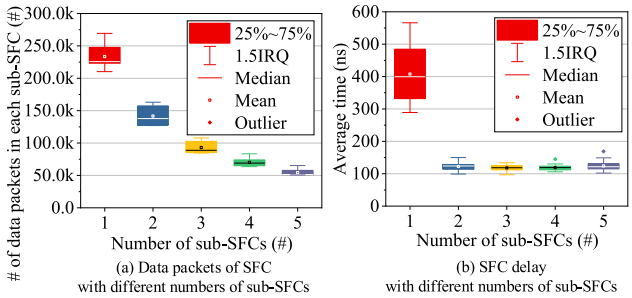


Fig. 10. a) data packets and b) SFC delay.

- Local VSCP (Local VNF Service Chain Placement) [19] keeps all VNFs as close as possible. The idea of Local VSCP is similar to the S1 placement strategy.
- Resource pool based algorithm [30] places the VNF chain over multiple servers based on different strategies (including centralized and distributed).
- RPA (Random Placement Algorithm) randomly places the VNF chain.
- HPA (Hybrid Placement Algorithm) places the VNF chain based on our proposed placement strategy.

C. Data Packets and SFC Delay

We evaluate the parallelized SFC consisting of only working sub-SFCs. We name the SFC containing one sub-SFC as the traditional SFC. As Fig. 10(a) shows, in the parallelized SFC, the more sub-SFCs, the fewer data packets. It means that the more sub-SFCs, the smaller flow after the split.

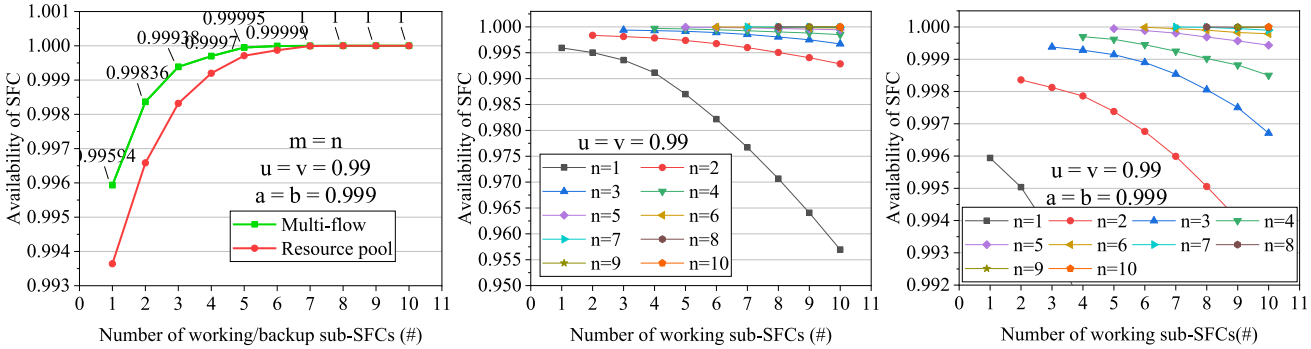
Fig. 10(b) shows the average delay of traditional SFC is 408 ns. For the parallelized SFC, when the number of sub-SFCs is 2 to 5, the average delays are distributed between 118 ns and 124 ns. We can conclude that in the parallelized SFC, the flow transmitted in each sub-SFC is small, resulting in a low SFC delay. In summary, the delay of parallelized SFC can be reduced by at least 30%.

In summary, the parallelized SFC can split large data flow into multiple small data flows. In this way, we can achieve high performance and reduce SFC delay. The average delay of parallelized SFC can be reduced by at least 30%.

D. Availability

We evaluate the availability of parallelized SFC with different numbers of sub-SFCs. Based on *Eq.* (18), we can conclude that both backup model and placement algorithm affect availability. In this experiment, we set $m \geq n$.

As Fig. 11(a) shows, we evaluate the availability of parallelized SFC with different numbers of sub-SFCs ($m = n$). We can observe that the availability of our proposed multi-flow backup model and placement algorithm is higher than the resource pool backup model and placement algorithm. The reason is that the resource pool backup model has more switches than the multi-flow backup model. In the resource pool backup model, the availability of switch needs to be considered [42]. Therefore, the multi-flow backup model outperforms the resource pool backup model. We can also observe that the availability increases as the number of sub-SFCs



(a) Availability of parallelized SFC with different working/backup sub-SFCs

(b) Availability of parallelized SFC with different working and backup sub-SFCs

(c) Detailed availability of parallelized SFC with different working and backup sub-SFCs

Fig. 11. Availability of parallelized SFC.

increases. When $m = 3$, the availability is 99.938%, satisfying three 9s. As for $m = 5$, it is 99.995%, satisfying four 9s. When $m \geq 6$, it satisfies five 9s or even higher. In summary, the multi-flow backup model can effectively improve the availability of parallelized SFC.

As Fig. 11(b) shows, we evaluate the availability of parallelized SFC with different numbers of working sub-SFCs and backup sub-SFCs. We can observe that under the same number of backup sub-SFCs, the less working sub-SFCs, the higher SFC availability. Under the same number of working sub-SFCs, the more backup sub-SFCs, the higher SFC availability. However, as the number of backup sub-SFCs increases, the effect of availability improvement is less obvious. For example, when m = 4, the availability is 99.12% when n = 1, 99.79% when n = 2, and 99.93% when n = 3. The first improvement is 0.67% while the second is 0.14%.

Fig. 11(c) shows part of the data of Fig. 11(b). We present a closer look at the detailed availability of parallelized SFC. For the sake of observation, we only show data with availability range of 0.992 to 1. In Fig. 11(c), we can determine the number of sub-SFCs based on the availability requirements. For example, if we want SFC availability to be no less than three 9s, we need to satisfy at least $m = n = 3$. However, the higher availability requirements, the more resource consumption. It is a trade-off between availability requirements and resource consumption.

In summary, the multi-flow backup model and hybrid placement algorithm can effectively improve the availability of parallelized SFC. Based on the delay, availability requirements, and resource constraints, we can adjust the number of working sub-SFCs and backup sub-SFCs.

E. PL Cost

As Fig. 12 shows, we evaluate PL cost in three cases. Firstly, we evaluate PL cost under different placement algorithms (Figs. 12 a and b). Secondly, PL cost of HPA under different placement strategies (Figs. 12 c and d). Thirdly, PL cost of HPA under different numbers of sub-SFCs (Figs. 12 e and f).

Since the size of DCN can effect PL cost, we evaluate PL cost with different number of PMs in DCN. Besides, when failures of working sub-SFCs occur, the switch redirects flow to backup VNF, resulting in extra PL cost. Therefore, we evaluate PL cost when running normally (Figs. 12 a, c, and e) and extra PL cost when failures occur (Figs. 12 b, d, and f).

1) PL Cost Under Different Placement Algorithms: Fig. 12(a) shows PL cost when working sub-SFCs running normally. We can observe that compared with RPA (baseline), Local VSCP and resource pool based placement algorithm can reduce PL cost by 20% and 40%, respectively. And our proposed HPA can reduce by nearly 60%. And Fig. 12(b) shows extra PL cost when failures occur. We can conclude that Local VSCP and resource pool based placement can reduce PL cost by 50% and 60%, respectively. And our proposed HPA can reduce by nearly 80%.

In summary, Local VSCP and resource pool cannot effectively reduce PL cost between different sub-SFCs. Therefore, HPA outperforms than Local VSCP and resource pool.

2) PL Cost Under Different Placement Strategies: In this subsection, we evaluate PL cost of HPA under the proposed three placement strategies.

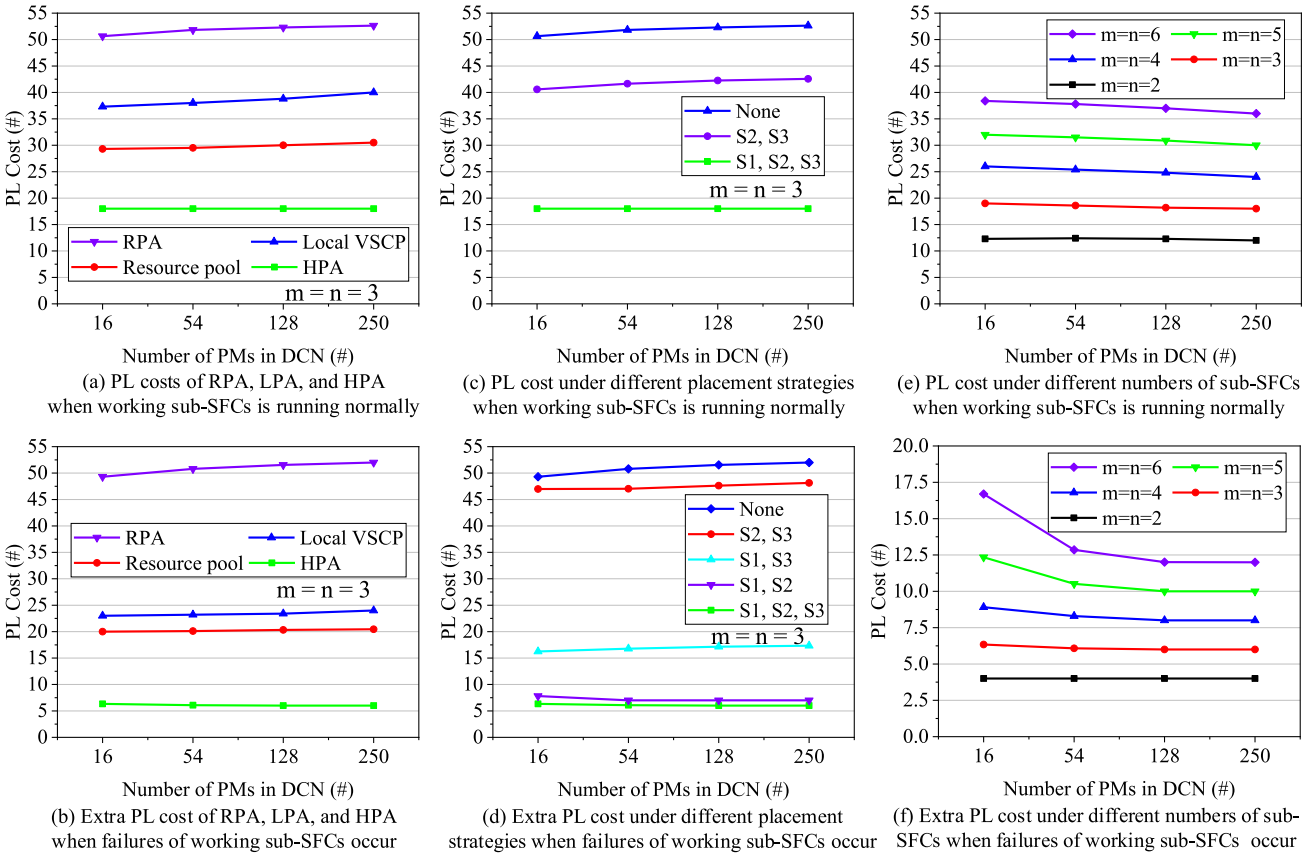
As Fig. 12(c) shows, we can observe that the three placement strategies can reduce PL cost by 60% when running normally. As Fig. 12(d) shows, the three placement strategies can reduce extra PL cost by 80% when failures occur. From Figs. 12(c) and 12(d), we can conclude that the optimization effect on PL cost is S1 > S2 > S3.

In summary, we can conclude that our proposed three placement strategies can significantly reduce PL cost (S1 > S2 > S3).

3) PL Cost Under Different Numbers of Sub-SFCs: In this subsection, we discuss PL cost of HPA under different numbers of sub-SFCs (including working sub-SFCs and backup sub-SFCs).

As Figs. 12(e) and 12(f) show, the less number of sub-SFCs, the lower PL cost, not only when running normally but also failures occur. Besides, PL cost tends to decline as the number of PMs increases, especially when failures occur. It means the more PMs in DCN, the more PMs in the same level of affinity domain. It is easier to place two VNFs in the same affinity domain, thus reducing PL cost.

In summary, when both running normally and failures occur, the less number of sub-SFCs, the lower PL cost. It is worth



(a) PL costs of RPA, LPA, and HPA when working sub-SFCs is running normally

(c) PL cost under different placement strategies when working sub-SFCs is running normally

(e) PL cost under different numbers of sub-SFCs when working sub-SFCs is running normally

(b) Extra PL cost of RPA, LPA, and HPA when failures of working sub-SFCs occur

(d) Extra PL cost under different placement strategies when failures of working sub-SFCs occur

(f) Extra PL cost under different numbers of sub-SFCs when failures of working sub-SFCs occur

Fig. 12. PL cost.

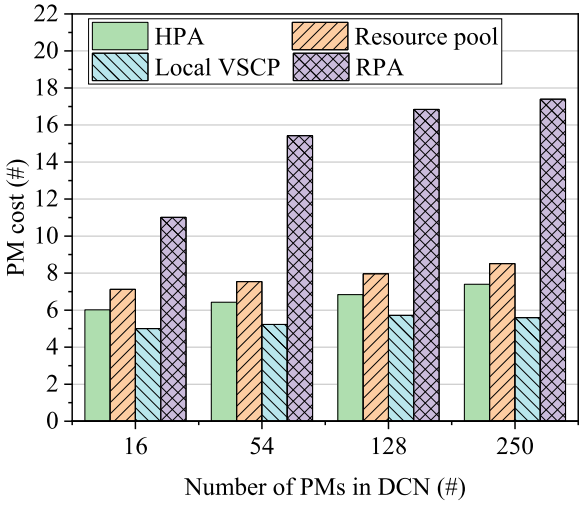


Fig. 13. PM costs of RPA, Resource pool, Local VSCP, and HPA.

mentioning that PL cost decreases as the number of PMs increases, especially when failures occur.

F. PM Cost

In this subsection, we evaluate PM cost of four placement algorithms including RPA (baseline), Resource pool [30], Local VSCP [19], and HPA (proposed).

As Fig. 13 shows, as the number of PMs increases, PM costs of HPA and Local VSCP do not change, while the resource pool increases gradually and RPA increases rapidly. This is because HPA uses PM by finding affinity domains and Local VSCP uses PM one by one, while other algorithms use PM in a greedy or random way. We can also observe that HPA outperforms than resource pool and RPA. It is worth mentioning that PM cost of Local VSCP is slightly smaller than HPA. It is because Local VSCP maps all VNFs in a local way, which keeps all VNFs as close as possible, thereby reducing more PM cost than HPA. However, keeping VNFs as close as possible can cause more VNFs to be unavailable when a single point of failure occurs in the PM. Therefore, HPA is more suitable than Local VSCP and resource pool for the parallelized SFC in terms of reducing PM cost.

In summary, HPA can effectively reduce PM cost. It is because HPA maps VNFs in the same sub-SFC based on high-affinity constraints (placement strategy S1). VNFs in the same sub-SFC are preferentially placed in the same PM, thus reducing PM cost.

G. Acceptance Ratio

To evaluate the performance of the placement algorithm, we define the acceptance ratio as follows:

$$Acceptance\ Ratio = \frac{\#\ of\ accepted\ SFC\ requests}{\#\ of\ total\ SFC\ requests} \quad (19)$$

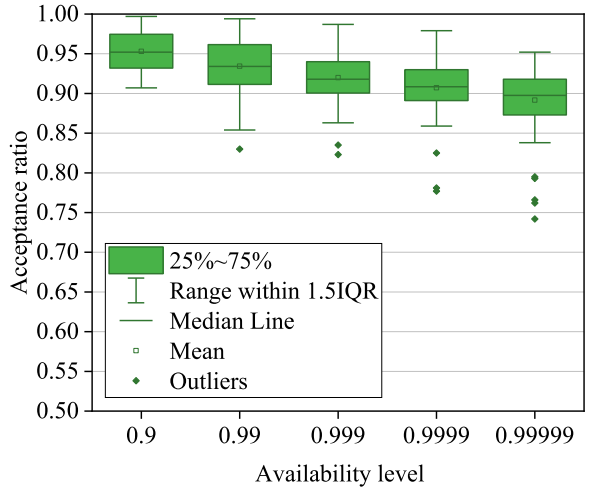


Fig. 14. Acceptance ratio of HPA.

The higher acceptance ratio, the better the placement algorithm in terms of resource optimization. We evaluate the acceptance ratio of HPA under five availability levels (one 9 $\sim$ five 9s).

We set up five test cases based on Fig. 11(a). And we set the number of SFC requests to 100 in each test case so that the acceptance rate can be obtained. For each test case, we set m = 1, 2, 3, 5, and 6 ($m \geq n$), corresponding to the five availability levels, respectively. For example, we set m = 3 and the availability level is three 9s. Then, in this experiment, we deploy each test case 100 times to confirm that the results are not affected by accidental events.

Fig. 14 shows that in most cases, the acceptance ratio of HPA can be achieved between 90% and 95%. However, the acceptance rate of HPA cannot reach 100% because the parallelized SFC placement problem has multiple resource constraints and performance requirements in addition to availability requirements. Also, there are some outliers when availability requirements are high. Because it might be more extreme cases when the requirements are higher. We can observe that the outliers increase as the availability level increases. From Fig. 14, we can also observe that as the availability level increases, the acceptance ratio of HPA decreases.

In summary, HPA can achieve a high acceptance ratio (90% to 95%). We can effectively improve availability (99.999%) by using the backup model and HPA.

VI. Conclusion

This article solves the parallelized SFC placement problem in DCN considering availability guarantee and resource optimization. Firstly, we define the parallelized SFC and propose a multi-flow backup model. The parallelized SFC improves data transmission efficiency and reduces SFC delay. And the backup model effectively improves the availability of parallelized SFC. Finally, we design three placement strategies and a hybrid placement algorithm to reduce PL cost. Compared with the related work, our proposed solutions can reduce SFC delay (30%) and optimize PL cost (reduce 40%) while guaranteeing availability (99.999%).

However, there are still some limitations. Firstly, the numbers of working and backup sub-SFCs are fixed in our solution. In future work, we plan to explore some potential solutions to support flexible numbers of working and backup sub-SFCs. Secondly, we do not consider the overhead required for NF state synchronization in the NFV execution model.

References

[1] (Oct. 2014). *Network Functions Virtualisation (NFV)*. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. 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, 1st Quart., 2016.

[3] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, Nov. 2016.

[4] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 407–420, Feb. 2017.

[5] J. Fan *et al.*, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, Oct. 2018.

[6] J. Gil-Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[7] S. G. Kulkarni *et al.*, "NFVnice: Dynamic backpressure and scheduling for NFV service chains," in *Proc. SIGCOMM*, 2017, pp. 71–84.

[8] J. Zhang, W. Wu, and J. C. S. Lui, "On the theory of function placement and chaining for network function virtualization," in *Proc. Mobihoc*, 2018, pp. 91–100.

[9] (Mar. 2019). *High Availability for OPNFV*. [Online]. Available: https://wiki.opnfv.org/display/availability

[10] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[11] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. SIGCOMM*, 2011, pp. 350–361.

[12] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, Oct. 2017.

[13] W. Ding, H.-F. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme," in *Proc. ICC*, 2017, pp. 1–6.

[14] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-aware backup allocation for a chain of virtual network functions," in *Proc. INFOCOM*, 2019, pp. 1918–1926.

[15] Z. Guo, W. Chen, Y. Liu, Y. Xu, and Z.-L. Zhang, "Joint switch upgrade and controller deployment in hybrid software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1012–1028, May 2019.

[16] G. Sallam and B. Ji, "Joint placement and allocation of virtual network functions with budget and capacity constraints," in *Proc. INFOCOM*, 2019, pp. 523–531.

[17] P. Sun, Z. Guo, J. Wang, J. Li, J. Lan, and Y. Hu, "DeepWeave: Accelerating job completion time with deep reinforcement learning-based coflow scheduling," in *Proc. IJCAI*, 2020, pp. 3314–3320.

[18] Z. Guo *et al.*, "AggreFlow: Achieving power efficiency, load balancing, and quality of service in data center networks," *IEEE ACM Trans. Netw.*, early access, Nov. 10, 2020, doi: 10.1109/TNET.2020.3026015.

[19] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstädter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Proc. GLOBECOM Workshops*, 2015, pp. 1–7.

[20] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, "SoftNIC: A software NIC to augment hardware," Dept. Electr. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Rep. UCB/EECS-2015-155, 2015.

[21] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "Netbricks: Taking the V out of NFV," in *Proc. OSDI*, 2016, pp. 203–216.

[22] G. P. Katsikas, T. Barbette, D. Kostic, R. Steinert, and G. Q. Maguire, "Metron: NFV service chains at the true speed of the underlying hardware," in *Proc. NSDI*, 2018, pp. 171–186.
[23] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: High performance and flexible networking using virtualization on commodity platforms," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 34–47, Mar. 2015.
[24] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. NSDI*, 2014, pp. 459–473.
[25] S. Palkar *et al.*, "E2: A framework for NFV applications," in *Proc. SOSP*, 2015, pp. 121–136.
[26] P. Zheng, A. Narayanan, and Z.-L. Zhang, "A closer look at NFV execution models," in *APNet*, 2019, pp. 85–91.
[27] H. Tang, D. Y. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 530–543, Mar. 2019.
[28] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "NFP: Enabling network function parallelism in NFV," in *Proc. SIGCOMM*, 2017, pp. 43–56.
[29] Y. Zhang *et al.*, "ParaBox: Exploiting parallelism for virtual network functions in service chaining," in *Proc. SOSR*, 2017, pp. 143–149.
[30] A. Engelmann and A. Jukan, "A reliability study of parallelized VNF chaining," in *Proc. ICC*, 2018, pp. 1–6.
[31] A. A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. INFOCOM*, 2013, pp. 2130–2138.
[32] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2249–2259, Oct. 2020.
[33] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *Proc. HotMiddlebox@SIGCOMM*, 2015, pp. 13–18.
[34] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. INFOCOM*, 2017, pp. 1–9.
[35] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Proc. IWQoS*, 2017, pp. 1–10.
[36] L. Qu, M. J. Khabbaz, and C. M. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, Mar. 2018.
[37] L. Qu, C. M. Assi, K. B. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 554–568, Sep. 2017.
[38] X. Meng, V. Pappas, and Y. Zhao, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. INFOCOM*, 2010, pp. 1154–1162.
[39] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. SIGCOMM*, 2008, pp. 63–74.
[40] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latré, and F. D. Turck, "Semantically enhanced mapping algorithm for affinity-constrained service function chain requests," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 317–331, Jun. 2017.
[41] S. Woo, J. Sherry, S. Han, S. B. Moon, S. Ratnasamy, and S. Shenker, "Elastic scaling of stateful network functions," in *Proc. NSDI*, 2018, pp. 299–312.
[42] Z. Guo, S. Dou, and W. Jiang, "Improving the path programmability for software-defined WANs under multiple controller failures," in *Proc. IEEE/ACM 28th Int. Symp. Qual. Service (IWQoS)*, 2020, pp. 1–10.
[43] (May 2015). *Berkeley Extensible Software Switch (BESS)*. [Online]. Available: http://span.cs.berkeley.edu/bess.html
[44] (Nov. 2020). *Data Plane Development Kit*. [Online]. Available: http://dpdk.org
[45] A. Fischer, J. Botero, M. Duelli, X. Hesselbach, and H. Meer, "Alevin—A framework to develop, compare, and analyze virtual network embedding algorithms," *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.*, vol. 37, pp. 1–12, Mar. 2011.
[46] (Aug. 2019). *Google Apps Service Level Agreement*. [Online]. Available: http://www.google.com/apps/intl/en/terms/sla.html

Meng Wang received the B.S. degree from Beijing Jiaotong University, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing. His current research interests include network function virtualization, network slicing, and resource allocation management.

Bo Cheng (Member, IEEE) received the Ph.D. degree in computer science from the University of Electronics Science and Technology of China in 2006. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include multimedia communications and services computing.

Shangguang Wang (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011, where he is an Associate Professor with the State Key Laboratory of Networking and Switching Technology. His research interests include service computing, cloud computing, and QoS management. He is the Vice Chair of IEEE Computer Society Technical Committee on Services Computing, the President of the Service Society Young Scientist Forum in China, served as the General Chair of CollaborateCom and ICCSA in 2016, and a TPC Chair of IOV and SC2 in 2014.

Junliang Chen is a Professor with the Beijing University of Posts and Telecommunications. His research interests are in the area of service creation technology. He was elected as a member of the Chinese Academy of Science in 1991, and a member of the Chinese Academy of Engineering in 1994.