Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy

Defang Li, *Student Member, IEEE*, Peilin Hong, Kaiping Xue, *Senior Member, IEEE*, and Jianing Pei

***Abstract*—By means of network function virtualization (NFV), dedicated proprietary network devices can be implemented as software and instantiated flexibly on common-off-the-shelf servers, in the form of virtual network functions (VNF). NFV can bring great cost reduction as well as operation flexibility. However, it also brings new problems, one of which is how to meet the availability of network services in the VNF deployment process, because of the error prone nature of software. The availability aware VNF deployment problem has attracted attention by academics, and reserving redundancy has been treated as the de facto technology. Compared with traditional backup schemes for physical machines, resource orchestration in NFV is more flexible and the characteristics of software should be considered to improve resource utilization efficiency. Based on the above considerations, in this paper we further study the availability aware VNF deployment problem in datacenter networks. To improve the resource utilization efficiency, the sharing mechanism of redundancy and multi-tenancy technology are taken into account. Then we formulate the problem mathematically and propose a joint deployment and backup scheme (JDBS). Finally, we conduct a numerical simulation in detail and compare it with four contrasting schemes in the existing literature. The simulation results show that JDBS is obviously superior to the contrasting schemes and can save about 40% resources at most.**

***Index Terms*—Network function virtualization, VNF deployment, SFC, availability aware, multi-tenancy, redundancy sharing.**

I. Introduction

NETWORK function virtualization (NFV) can transform traditional network devices into software and instantiate them on common-off-the-shelf servers [1], [2], which enhances the flexibilities and conveniences of cloud services. By leveraging this new technology, many small and medium businesses can outsource their IT infrastructures to the cloud, which can save a great of capital expenditures and operating expenses (CAPEX/OPEX) as a result [3], [4]. For cloud service providers (CSP), NFV improves their resource utilization efficiency and management flexibility. Generally, network services customized by users are carried out by service function chains (SFC) in NFV, each of which is composed of a series of ordered virtual network functions (VNF). Thus, for simplicity in our work, network service requests are all treated as SFC requests (SFCR).

To obtain satisfactory quality of experience, the availability of each service should be guaranteed. Meanwhile, the software nature of VNF brings both flexibilities and challenges on the study of availability assurance in NFV.

In this paper, we study the VNF deployment problem in a datacenter while guaranteeing the availability requirements of different SFCRs, which is also known as the availability aware VNF deployment problem [5], [6]. To meet these availability requirements of different SFCRs, reserving redundancy is treated as the de facto technology in some existing literatures, such as [5], [7], [8]. However, compared with backup schemes for physical machines, three following factors should be highlighted when reserving redundancy in NFV.

A. BRC and Multi-Tenancy

Generally, the resource consumption of a VNF can be classified into two parts: One part is the virtualization overheads [9] when instantiating VNFs, e.g., the resource consumption to maintain the image and related libraries of a VNF, which is called basic resource consumption (BRC) in our previous work [10]. The other part is used to accomplish service when the VNF is in operation. To distinguish from BRC, we name it as Duty Resource Consumption (DRC).

Virtual machine (VM) and container are two popular virtualization technologies recently, and they have their own advantages and disadvantages [11]. In this paper, VNFs are assumed to be implemented in VMs. From the research in [9], instantiating more VMs on a server will incur more BRC. To reduce BRC, multi-tenancy technology, one kind of software structure, can be utilized to make multiple tenants share the same software instance.

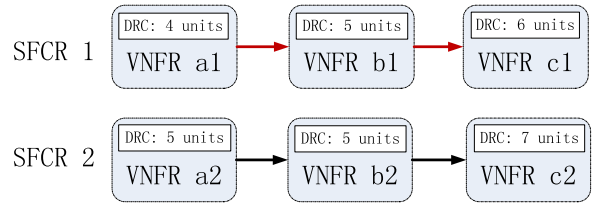
Compared with single-tenancy architecture, by which each VNFR has its own dedicated VNF instance, multiple VNFRs of the same type can be hosted on the same VNF instance with multi-tenancy technology. So multi-tenancy technology can save BRC by reducing the number of instantiated VNFs.

Fig. 1 gives an example. Fig. 1(a) shows 2 SFCRs and their DRC demands are labelled on the top left corner respectively. Fig. 1(b) illuminates the VNFs serving above 2 SFCRs, and these VNFs are instantiated in multi-tenancy principle. From the figure, we can see that an SFC can serve more than one

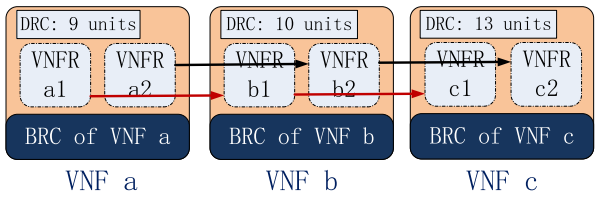
Manuscript received May 6, 2019; revised August 2, 2019; accepted August 14, 2019. Date of publication August 20, 2019; date of current version December 10, 2019. The work is supported in part by National Natural Science Foundation of China (NSFC) under Grants No.61671420 and No. 61672484, and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. 2016394. The associate editor coordinating the review of this article and approving it for publication was D. Hutchison. *(Corresponding author: Peilin Hong.)*

The authors are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: ldf911@mail.ustc.edu.cn; plhong@ustc.edu.cn; kpxue@ustc.edu.cn; jianingp@mail.ustc.edu.cn).

Digital Object Identifier 10.1109/TNSM.2019.2936505



(a) Two SFCRs



(b) Multi-tenancy based VNFs and the formed chains for above two SFCRs

Fig. 1. SFCRs and multi-tenancy based VNFs.

SFCR based on multi-tenancy technology. For each of the multi-tenant capable VNF, BRC is fixed and DRC is the linear accumulation of VNFRs' DRC on it.

B. Redundancy Sharing

Traditionally, there are two kinds of backup schemes: 1:1 scheme and M:N scheme [7], [12], [13]. In 1:1 scheme, each primary entity has a dedicated backup entity, and both of these two entities have fixed BRC and DRC, which has a good disaster recovery capability. However, the availability requirements are not so high for some services and the 1:1 backup scheme is resource inefficient in this scenario. In M:N scheme, there are M primary entities and N backup entities, where N is smaller than M generally. Usually, one backup entity is responsible for one specific primary entity. In other words, these backup entities are dedicated. However, in the asynchronous backup scenario where all primary entities have similar functioning, these *N* backup entities can dynamically provide backups for the *M* primary entities together and any backup entity can be selected to replace any primary entity [5].

In this paper, we combine the above two schemes to redesign the backup scheme in our work. Firstly, a backup VNF is reserved for each primary VNF. However, all these backup VNFs work in active-standby mode [14], in which we just allocate BRC to each backup VNF. According to ETSI (European Telecommunication Standards Institute)'s technical report [15], resources can be allocated to VNFs on demand based on NFV-MANO (NFV Management and Orchestration). So DRC of backup VNFs can be pooled and shared. Each backup VNF can communicate with its responsible primary VNF timely to obtain its working status in the active-standby mode, and replace the failed primary VNF immediately when faults are detected. Compared with dedicated redundancy, shared redundancy can improve the availability in a resource-efficient manner [6].

C. Tradeoff Between BRC and Shared DRC Redundancy

With the help of multi-tenancy, VNFRs from different SFCRs can be hosted on the same VNF instance. Furthermore, these VNFRs have the same service availability, which is the availability of their host VNF.

Consider the following example: there are 6 VNFRs of the same type; each one needs 5 units of DRC; the availability of a VNF without backup is 0.999 and the availability of each backup VNF is also 0.999. If we instantiate 3 VNFs, and each of which has 2 VNFRs on it, then each VNF needs 10 units of DRC. With 20 units of shared DRC redundancy for above 3 VNFs, the availability of each VNF will be $0.999 + 0.001 \cdot (C_2^0 0.001^0 \cdot 0.999^2 + C_2^1 0.001 \cdot 0.999) \cdot 0.999 = 0.999998$ (The detailed calculating method is shown in Section III-C). If we instantiate 6 VNFs, and each of which has 1 VNFR on it, then each VNF demands 5 units of DRC. With 15 units of shared DRC redundancy for above 6 VNFs, the availability of each VNF will be $0.999 + 0.001 \cdot (C_5^0 0.001^0 \cdot 0.999^5 + C_5^1 0.001^1 \cdot 0.999^4 + C_5^2 0.001^2 \cdot 0.999^3) \cdot 0.999 = 0.999999$.

Therefore, for the same set of VNFRs, the more VNFs are instantiated, the less shared DRC redundancy is needed to get the same availability improvement. However, more VNFs result in more BRC. So there is a tradeoff between BRC and shared DRC redundancy, which should be well addressed to reduce the total resource consumption.

In summary, given a set of SFCRs, we need to deploy a series of VNFs to serve them in the datacenter, while assuring that the availability requirements of different SFCRs are satisfied. To improve the resource utilization efficiency, the redundancy sharing mechanism and multi-tenancy technology are introduced. The problem of availability aware VNF deployment is NP-hard, and a joint deployment and backup scheme, JDBS, is proposed. Our major contributions can be summarized as below:

- We consider the software characteristics of VNFs in the availability aware VNF deployment problem and clarify BRC and DRC when instantiating a VNF. To improve the resource utilization efficiency, the sharing mechanism of redundancy and multi-tenancy technology are introduced.
- We design an availability calculation algorithm for the shared redundancy based backup scheme, to calculate modified availability of a VNF.
- We formulate the availability aware VNF deployment problem mathematically, and propose a joint deployment and backup scheme, JDBS. Then the performance of JDBS is compared with solutions in existing literatures. The simulation results show that JDBS outperforms the contrasting schemes and can save about 40% resources at most.

The remainder of our paper is organized as follows. A literature review is given in Section II. Then we state and formulate the problem in Section III, and the proposed solution is described in Section IV. After that, Section V demonstrates the simulation results. Section VI concludes our work. Finally, we make a discussion about the significance of our research in Section VII.

II. Related Work

A. General VNF Deployment

Mathematically, VNF deployment in the datacenter networks is usually formulated as an integer linear programming (ILP) [16], [17], mixed integer linear programming (MILP) [18], or mixed integer quadratically constraints programming (MIQCP) problem [19], which is NP-hard in general. From an optimization perspective, the object can be minimizing utilized physical machines [20], minimizing the total resource consumption [16], [21], [22] or minimizing the total service delay [18]. Mehraghdam *et al.* [19] made a discussion about all above three optimization objects.

Furthermore, many researchers considered the VNF placement problem with other features jointly. Luizelli *et al.* [17] formalized the network function placement and chaining problem, then proposed an ILP model to solve it. It is noteworthy that VNFs are geo-distributed in their model. Pham *et al.* [23] studied the VNF placement problem for SFCs with the purpose of energy saving and traffic-aware cost minimization. Ye *et al.* [24] considered the problem that how to jointly optimize the topology design and mapping of multiple SFCs so as to minimize the total bandwidth consumption.

Different from traditional VNF deployment problem, we not only consider the deployment of VNFs for a given set of SFCRs, but also solve how to meet the availability requirements of these SFCRs. As a result, non-linear constraints are introduced when formulating the problem mathematically. Moreover, when setting the optimization object, we need to strike a balance between nodes' resource consumption and bandwidth consumption.

B. Availability Aware VNF Deployment

Recently, how to guarantee availabilities of NFV based services have been attracting researchers' attention. ETSI's reports give clear descriptions and requirements about reliability/availability of VNF and SFC [7], [13]. Especially in [7], the reliability/availability model and principles about how to assure the end-to-end reliability/availability are demonstrated clearly.

Liu *et al.* [25] proposed a framework to evaluate the reliability of the NFV deployment, and they described 4 algorithms to solve the minimum total failure removal problem. Meanwhile, Di Mauro *et al.* [26] made an availability evaluation of an SFC using reliability block diagram and stochastic reward networks. These two work focus on the availability evaluation, rather than the specific solution to improve the availability of VNFs. However, they gave the guidance on how to design a reliable deployment scheme.

Kang *et al.* [5] investigated the trade-off between end-to-end reliability and computational load per server via the joint design of VNF chain composition and forwarding graph embedding, under the assumption of a bipartite forwarding graph that consists of controller and regular VNFs. Taleb *et al.* [27] introduced a framework, along with efficient and proactive restoration mechanisms, to ensure service resilience of the 5G mobile system. Their focus is on the MME (mobility management entity) VNF failure restoration process, not the backup scheme. Fan *et al.* [8] presented a novel online algorithm to minimize the physical resource consumptions and meanwhile guarantee the required high reliability by reserving redundancy for the most unreliable VNFs. Moreover, they also proposed a series of methods to allocate backup resource in order to maximize the number of SFCRs that can be served, while meeting their heterogeneous availability requirements in [28]. Ding *et al.* [29] optimized the solution in [8] by providing backups for the VNFs with the largest cost-aware importance measure (CIM).

Although the above-mentioned solutions [5], [8], [28], [29] are effective to their studied problems, the resource utilization efficiency of these solutions is low relatively. Because one backup VNF is corresponding to one particular primary VNF, which cannot provide backup for other primary VNFs with potential failures.

Kanizo *et al.* [30] claimed that a significant cost reduction can be achieved through resource sharing among different VNFs. They applied an M:N backup scheme in their solution and gave a theoretical analysis about the NP-hardness of the problem. However, they did not consider VNFs as chains, so the inter-relationships between the VNFs are ignored. Several measures on how backup resources can be integrated into the embedding of VNFs are discussed in [31], and a shared redundancy based resource allocation algorithm is proposed. Qu *et al.* [32] jointly maximized the achieved respective reliability of supported network services and minimized these services' respective end-to-end delays. Furthermore, they studied the reliability-aware joint VNF chain placement and flow routing optimization in [6], and the backup VNF can be shared by the adjacent VNFs to improve the resource utilization efficiency.

Compared with existing researches, our proposed backup scheme is the combination of the dedicated redundancy based and shared redundancy based backup schemes. In our proposed solution, we reserve backup VNF for every primary VNF to support the active-standby mode backup scheme. So BRC of each backup VNF is reserved in our solution. DRC redundancy can be allocated to the backup VNFs on demand, and it is pooled and shared by multiple backup VNFs. Besides above differences, multi-tenancy technology is also considered in our solution to improve the resource utilization efficiency further.

C. Multi-Tenancy and BRC

Multi-tenancy is a software architecture in the realm of software as a service (SaaS) business model [33], which allows multiple tenants to share the same software instance. Compared with single-tenant architecture, in which each tenant gets its own application instance, multi-tenant architecture can provide higher resource utilization, lower service price, and more efficient management for CSPs [34]. Medhat *et al.* [35] pointed that mobile network operators could share VNFs while maintaining separate logical data and control planes, and a framework is evaluated utilizing open-source tools and virtual tenant network techniques. In our paper, multi-tenancy

technology is applied to the implementation of VNFs to reduce the number of VNF instances.

VNFs are network functions implemented in software and usually run on VMs. As is mentioned in [36], [37], virtualization or instantiating a software needs extra costs, also known as virtualization overheads. Chen *et al.* [9] claimed that virtualization introduces an additional layer of abstraction that produces resource overheads, which should be considered in the resource allocation process. Specifically, they made a detailed measurements about the overheads of some popular virtualization technologies. For a VNF, it needs at least an image, an OS, some related libraries and a virtualization layer to support its regular functioning. So it consumes some resources to support its regular functioning even in idle state, and we call this part of resource consumption BRC. We considered BRC in our previous work [10], which deals with the resource-efficient VNF placement in a cloud datacenter. Wen *et al.* [38] claimed that a server will consume some resources to instantiate a VNF, which is very similar to the BRC in our model.

III. Problem Statement and Formulation

A. Network Model

The substrate network is represented as an undirected graph $G = (N^s, E^s)$, where N^s indicates the set of total nodes in the substrate network and E^s indicates the link set of the substrate network. Specifically, *P* is used to indicate the set of total servers.

B. Availability Model

1) Availability Model for Single VNF: VNFs are repairable entities that can be fixed by software updating or reloading. Assuming that both operational and repair time is governed by a Markov chain, then we can use the following equation to calculate the availability of a VNF [7]:

$$p = \frac{\text{uptime}}{\text{downtime} + \text{uptime}} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}, \quad (1)$$

where uptime is the time that VNF is in operation regularly, which is also known as mean time between failures (MTBF), and downtime is the time that VNF is out of service, which is also known as mean time to repair (MTTR).

2) Availability Model for an SFC: Link and switch failures are not considered in this paper, as modern datacenters typically have rich path diversity between any pair of servers [39], [40], which can effectively resist these failures. So only VNF failures are considered. Usually, the faults of VNFs are independent of each other, so the availability of SFC γ can be calculated as:

$$p_\gamma = \prod_{i=1}^{n} p_i, \quad (2)$$

where *n* is the number of VNFs in SFC γ, and p_i is the availability of VNF *i* contained in SFC γ.

Algorithm 1: Modify Availability

Input: Number of VNFs: $\mathbb{N}$,
Shared DRC Redundancy: R_d;
Output: Modified availability of $\mathbb{N}$ VNFs.
for *i in* $\mathbb{N}$ **do**
  $q^i = p^i$.
  $R_\delta = R_d - r^i$.
  $\vec{l_r} = (r^1, r^2, \ldots, r^{i-1}, r^{i+1}, \ldots, r^{\mathbb{N}})$.
  Calculate the Cartesian power of $\mathbb{N} - 1$ {0,1} sets, indicated as Ω.
  for $\vec{\omega}$ *in* Ω **do**
    $R_\omega = \vec{l_r} \cdot \vec{\omega}^T$.
    if $R_\delta \geq R_\omega$ **then**
      Calculate the probability of working status ω, indicated as p^ω.
      $q^i = q^i + p^{\text{bk}} \cdot (1 - q^i) \cdot p^\omega$.
    else
      Continue.
    end
  end
end

C. Availability Modification of VNFs Based on Shared Redundancy

Algorithm 1 is designed to figure out the modified availabilities of the VNFs based on a given block of shared DRC redundancy.

In Algorithm 1, p^i indicates the modified availability of VNF *i*, q^i indicates the availability of VNF *i* without any redundancy (It can be also called the inherent availability.), r^i indicates the DRC demand by VNF *i*, R_δ indicates the residual DRC redundancy after reserving redundancy for VNF *i*, and R_ω indicates the needed DRC redundancy to handle failures of other VNFs except VNF *i*. We expound Algorithm 1 through a simple example as follows:

Assume that there are 3 VNFs, namely VNF *e*, VNF *f*, and VNF *g*. Their DRCs are $r^e = 3$ units, $r^f = 4$ units, and $r^g = 5$ units with inherent availability of $p^e = 0.94$, $p^f = 0.95$ and $p^g = 0.96$ respectively. The availabilities of all backup VNFs are assumed to be $p^{\text{bk}} = 0.95$. Then we consider the availability modification of VNF *e* with 8 units of shared DRC redundancy, that is to say, $R_d = 8$ units.

Firstly, we need to set up a {0, 1} set for VNF *f* and VNF *g* respectively. The value 0 indicates that the corresponding VNF is functioning regularly, and the value 1 indicates that one or more faults happen on the VNF. For the 2 VNFs, we calculate the Cartesian power [41] of the two corresponding {0, 1} sets, where the result is {(0, 0), (0, 1), (1, 0), (1, 1)}. Each tuple in {(0, 0), (0, 1), (1, 0), (1, 1)} indicates one working status of VNF *f* and VNF *g*.

Then the availability improvement of VNF *e* is calculated in each working status based on 8 units of shared DRC redundancy, which is shown in Table I. The parameter I_f indicates the working status of VNF *f*, and I_g indicates the working status of VNF *g*. In Table I, $R_\delta = R_d - r^e = 5$ units. The

TABLE I
AVAILABILITY MODIFICATION CASE OF VNF e

I_f	I_g	$R_\delta \geq R_{fg}$	p^e_δ
0	0	1	0.051984
0	1	1	0.000216
1	0	1	0.000274
1	1	0	0

parameter R_{fg} is the needed DRC redundancy to handle the failures of VNF f and VNF g, which is calculated by Eq. (3). The availability improvement of VNF e in each working status, indicated as p^e_δ, can be calculated by Eq. (4).

$$R_{fg} = I_f \cdot r^f + I_g \cdot r^g. \tag{3}$$

$$p^e_\delta = (1-p^e)\cdot[(1-I_e)\cdot p^f + I_f\cdot(1-p^f)] \times [(1-I_g)\cdot p^g + I_g\cdot(1-p^g)]\cdot p^{\mathrm{bk}}. \tag{4}$$

After calculating the availability improvements of all working statuses, the modified availability of VNF e given 8 units of shared DRC redundancy is $q^e = p^e + 0.051984 + 0.000216 + 0.000274 + 0 = 0.992474$.

Similarly, the modified availability of the other 2 VNFs can be derived.

D. Problem Statement

In this paper, we study the availability aware VNF deployment problem given a set of SFCRs. A series of VNFs need to be deployed in the datacenter to serve them, while redundancy should be reserved to meet the availability requirements of these SFCRs. To improve the utilizing efficiency of network resource, the redundancy sharing mechanism and multi-tenancy technology are considered. As stated in Section I-C, for the same set of VNFRs, the more VNFs are instantiated, the greater availability improvement can be obtained based on the same block of shared DRC redundancy. However, BRC will be more as a result. If the VNFRs are hosted on a smaller set of VNFs utilizing multi-tenancy technology, BRC will be less. However, the required DRC by each primary VNF will be larger. Then the shared redundancy may increase in order to get the same availability improvement. So there exists a balance between BRC and shared DRC redundancy. Thus it is crucial to decide which VNFRs should be hosted on a multi-tenant capable VNF instance.

In our availability aware VNF deployment problem, the following questions need to be solved.

- **Q1:** How to coordinate the relationship between deployment and backup.
- **Q2:** Which VNFRs should be hosted on a multi-tenant capable VNF.
- **Q3:** Where to place the redundancy, and how much the redundancy should be.

In this paper, we assume that VNFs are implemented utilizing VM. Each VNF is corresponding to an isolated VM.

TABLE II
NOTATIONS

Parameters	Descriptions
Topology related	
N^s	set of total nodes in the substrate network.
P	set of total servers.
E^s	link set of the substrate network.
n^s_u, n^s_v	two nodes in the substrate network.
(n^s_u, n^s_v)	substrate link between n^s_u and n^s_v.
Resource related	
$\mathrm{cpu}_{\gamma,n^\nu_i}$	CPU DRC by VNFR n^ν_i in SFCR γ.
$\mathrm{mem}_{\gamma,n^\nu_i}$	memory DRC by VNFR n^ν_i in SFCR γ.
$\mathrm{brc}^{\mathrm{cpu}}_\phi$	CPU BRC when instantiating a VNF ϕ, $\phi \in \Phi$.
$\mathrm{brc}^{\mathrm{mem}}_\phi$	memory BRC when instantiating a VNF ϕ, $\phi \in \Phi$.
$b_{\gamma,n^\nu_i,n^\nu_j}$	bandwidth consumption by logical link $(n^\nu_i\ n^\nu_j)$ in SFCR γ.
$C^{\mathrm{cpu}}_{n^s_u}$	available CPU capacity of server n^s_u.
$C^{\mathrm{mem}}_{n^s_u}$	available memory capacity of server n^s_u.
$C^{\mathrm{link}}_{(n^s_u,n^s_v)}$	available bandwidth of substrate link (n^s_u, n^s_v).
Binary variables	
$x_{\gamma,n^\nu_i,\lambda}$	whether VNFR n^ν_i in SFCR γ is hosted on VM λ. If yes, $x_{\gamma,n^\nu_i,\lambda} = 1$, otherwise, 0.
$l_{\gamma,n^\nu_i,\phi}$	whether VNFR n^ν_i in SFCR γ demands VNF ϕ. If yes, $l_{\gamma,n^\nu_i,\phi} = 1$, otherwise, 0.
$k_{\lambda,\phi}$	whether VM λ is implemented as an instance of VNF ϕ, and tuple (λ,ϕ) can indicate an unique VNF. If yes, $k_{\lambda,\phi} = 1$, otherwise, 0.
z_{λ,ϕ,n^s_u}	whether VNF (λ,ϕ) is deployed on server n^s_u, $n^s_u \in P$. If yes, $z_{\lambda,\phi,n^s_u} = 1$, otherwise, 0.
$y_{\gamma,n^\nu_i,n^\nu_j,n^s_u,n^s_v}$	whether the mapping path of logical link (n^ν_i, n^ν_j) in SFCR γ goes through substrate link (n^s_u, n^s_v). If yes, $y_{\gamma,n^\nu_i,n^\nu_j,n^s_u,n^s_v} = 1$, otherwise, 0.
$\zeta_{\lambda,\phi,n^s_u}$	whether the backup of VNF (λ,ϕ) is placed on server n^s_u, $n^s_u \in P$. If yes, $\zeta_{\lambda,\phi,n^s_u} = 1$, otherwise, 0.
SFCR related	
Γ	set of total SFCRs, $\gamma \in \Gamma$ indicates an SFCR.
Ψ_γ	set of total VNFRs in SFCR γ.
E^ν_γ	set of logical links between the VNFRs in SFCR γ.
n^ν_i, n^ν_j	two nodes in an SFCR.
(n^ν_i, n^ν_j)	logical link between n^ν_i and n^ν_j, $(n^\nu_i, n^\nu_j) \in E^\nu_\gamma$.
Λ	set of VMs, $\lambda \in \Lambda$ indicates a VM.
Φ	number of VNF types, $\phi \in \{0, 1, ..., \Phi-1\}$.
$p^{\lambda,\phi}$	inherent availability of VNF (λ,ϕ).
q^{λ,ϕ,n^s_u}	modified availability of VNF (λ,ϕ), the backup of which is on server n^s_u.
p^{bk}	availability of a backup VNF.
A_γ	availability requirement of SFCR γ.

E. Problem Formulation

In this part, we make a formulation about the availability aware VNF deployment problem. The main notations are listed in Table II.

Firstly, constraints about the binary variables are demonstrated.

For $x_{\gamma,n^\nu_i,\lambda}$:

$$\sum_{\lambda\in\Lambda} x_{\gamma,n^\nu_i,\lambda} = 1, \tag{5}$$

Here, Eq. (5) restricts that a VNFR should be hosted on one and only one VM (a multi-tenant capable VNF). Moreover,

to limit the mutual interference among different VNFRs in a reasonable degree, the number of VNFRs in the same VM cannot exceed a threshold:

$$\sum_{\gamma\in\Gamma}\sum_{n_i^\nu\in\Psi_\gamma} x_{\gamma,n_i^\nu,\lambda} \le \eta, \tag{6}$$

η indicates the maximum number of VNFRs that a VM can host.

For $k_{\lambda,\phi}$:

$$k_{\lambda,\phi} = \begin{cases} 1, & \sum_{\gamma\in\Gamma}\sum_{n_i^\nu\in\Psi_\gamma} x_{\gamma,n_i^\nu,\lambda}\cdot l_{\gamma,n_i^\nu,\phi} \ge 1; \\ 0, & \text{otherwise.} \end{cases}, \tag{7}$$

Eq. (7) indicates that if more than one VNFR demanding for VNF ϕ is hosted on VM λ, the VM will become an instance of VNF ϕ. Furthermore, a VM can only be instantiated as one kind of VNF at most, so:

$$\sum_{\phi\in\Phi} k_{\lambda,\phi} \le 1, \tag{8}$$

For z_{λ,ϕ,n_u^s}:

$$\sum_{n_u^s\in P} z_{\lambda,\phi,n_u^s} = 1, \tag{9}$$

Here, Eq. (9) restricts that a VNF instance should be hosted on one and only one server.

For a given set of SFCRs, considering the tradeoff between bandwidth optimization and nodes' resource optimization, the optimization object is set as minimizing the number of used servers:

$$\min \quad \sum_{n_u^s\in P} h_{n_u^s}, \tag{10a}$$

$$h_{n_u^s} = \begin{cases} 1, & \sum_{\lambda\in\Lambda}\sum_{\phi\in\Phi} z_{\lambda,\phi,n_u^s} \ge 1; \\ 0, & \text{otherwise.} \end{cases}, \tag{10b}$$

Eq. (10b) indicates that if more than one VNF instance is mapped on server n_u^s, the server has to be activated.

Nextly, nodes' resource constraints are introduced in. To simplify the system model, we assume that BRC of each VNF is fixed, and it is accumulated linearly when multiple VNFs are placed on the same server. Then the CPU constraints are as follows:

$$R_{\mathrm{drc}}^{\mathrm{pri,cpu}}(\lambda,\phi) = \sum_{\gamma\in\Gamma}\sum_{n_i^\nu\in\Psi_\gamma} x_{\gamma,n_i^\nu,\lambda}\cdot l_{\gamma,n_i^\nu,\phi}\cdot \mathrm{cpu}_{\gamma,n_i^\nu}, \tag{11a}$$

$$R_{\mathrm{brc}}^{\mathrm{pri,cpu}}(\lambda,\phi) = \sum_{\gamma\in\Gamma}\sum_{n_i^\nu\in\Psi_\gamma} x_{\gamma,n_i^\nu,\lambda}\cdot l_{\gamma,n_i^\nu,\phi}\cdot \mathrm{brc}_\phi^{\mathrm{cpu}}, \tag{11b}$$

$$R_{\mathrm{brc}}^{\mathrm{bk,cpu}}(n_u^s) = \sum_{\lambda\in\Lambda}\sum_{\phi\in\Phi}\zeta_{\lambda,\phi,n_u^s}\cdot \mathrm{brc}_\phi^{\mathrm{cpu}}, \tag{11c}$$

$$\sum_{\lambda\in\Lambda}\sum_{\phi\in\Phi}(R_{\mathrm{drc}}^{\mathrm{pri,cpu}}(\lambda,\phi)+R_{\mathrm{brc}}^{\mathrm{pri,cpu}}(\lambda,\phi))\cdot z_{\lambda,\phi,n_u^s} + R_{\mathrm{brc}}^{\mathrm{bk,cpu}}(n_u^s)+R_{\mathrm{drc}}^{\mathrm{bk,cpu}}(n_u^s) \le C_{n_u^s}^{\mathrm{cpu}}, \tag{11d}$$

where $R_{\mathrm{drc}}^{\mathrm{pri,cpu}}(\lambda,\phi)$ and $R_{\mathrm{brc}}^{\mathrm{pri,cpu}}(\lambda,\phi)$ are CPU DRC and CPU BRC by the VNFRs mapped on primary VNF (λ,ϕ), respectively. The parameters $R_{\mathrm{brc}}^{\mathrm{bk,cpu}}(n_u^s)$ and $R_{\mathrm{drc}}^{\mathrm{bk,cpu}}(n_u^s)$ are CPU BRC and shared CPU DRC redundancy by backup VNFs on server n_u^s, respectively. Figuring out $\zeta_{\lambda,\phi,n_u^s}$ and $R_{\mathrm{drc}}^{\mathrm{bk,cpu}}(n_u^s)$ on each server are the keys of availability aware VNF deployment problem in this paper.

Similarly, we can get the memory resource constraints.

After setting constraints on nodes' resource, the bandwidth constraint of each link is as follows:

$$\sum_{\gamma\in\Gamma}\sum_{(n_i^\nu,n_j^\nu)\in E_\gamma^\nu} b_{\gamma,n_i^\nu,n_j^\nu}\cdot y_{\gamma,n_i^\nu,n_j^\nu,n_u^s,n_v^s} \le C_{(n_u^s,n_v^s)}^{\mathrm{link}}, \tag{12}$$

Finally, we should guarantee that the availability requirement of each SFCR is satisfied. The availability of SFCR γ is determined by the VNFs serving it, and for n_i^ν in SFCR γ, its availability is:

$$p^{\gamma,n_i^\nu} = \sum_{\lambda\in\Lambda}\sum_{\phi\in\Phi}\sum_{n_u^s\in P} x_{\gamma,n_i^\nu,\lambda}\cdot k_{\lambda,\phi}\cdot\zeta_{\lambda,\phi,n_u^s}\cdot q^{\lambda,\phi,n_u^s}, \tag{13}$$

q^{λ,ϕ,n_u^s} is the modified availability of VNF (λ,ϕ) whose backup is on server n_u^s. It is the function of shared DRC redundancy, number of primary VNF instances that share the same block of redundancy, the inherent availability and DRC demand of these VNF instances. It is complicated, and we cannot give an explicit mathematical expression for now. However, we have designed an algorithm to get q^{λ,ϕ,n_u^s} exactly given above-mentioned variables, which is Algorithm 1.

Then for SFCR γ, its availability requirement should be met:

$$\prod_{n_i^\nu\in\Psi_\gamma} p^{\gamma,n_i^\nu} \ge A_\gamma. \tag{14}$$

The VNF deployment problem is NP-hard [20], [21], which is a sub-question of our problem. So our problem is NP-hard too. Besides, there are non-linear constraint (Eq. (14)) and non-explicit relationship (q^{λ,ϕ,n_u^s}) in the formulations. So it is incapable to solve the problem in theory but practical to design an efficient heuristic solution.

IV. Proposed Solution

A. Framework of JDBS

To solve the questions in Section III-D, we propose a Joint Deployment and Backup Scheme, JDBS.

In our solution, the processes of VNF deployment and backup run alternately based on servers. Specifically, a server is filled with SFCRs (or VNFRs) as much as possible firstly. Then related VNF instances are deployed on the server; the shared DRC redundancy and BRC are reserved in its neighbor. The above processes solve **Q1** and the former part of **Q3**.

As stated before, for the same set of VNFRs, the more VNFs are instantiated, the less shared redundancy is needed to get the desired availability improvement. However, BRC increase with the increasing number of VNF instances. So which and how many VNFRs should be hosted on the same multi-tenant capable VNF instance should be addressed carefully, which will determine the number of VNFs and the DRC demand of each VNF.

In our solution, an iterative process is designed to make a balance between BRC and shared DRC redundancy. The

Algorithm 2: Joint Deployment and Backup Scheme, JDBS

1:**Input**: Γ, (N^s, E^s)
2:**Output**: Mapping results of Γ to (N^s, E^s)
3:Sort all SFCRs in descending order based on their resource demands.
4:$i = 0$.
5:**while** Γ *is not empty* **do**
6: Start a new server, $i += 1$.
7: Map SFCRs into server *i*, *map sfcrs*().
8: Establish routing paths of flows using the shortest path algorithm.
9: Reserve redundancy for SFCRs in server *i*, *reserve redundancy*().
10: **while** *1* **do**
11: **if** *there are VNFRs that can be merged* **then**
12: Merge together VNFRs demanding the same type of VNF in server *i*, *merge vnfrs*().
13: **else**
14: Break.
15: **end**
16: **if** *SFCR with least resource demand can be hosted on server i* **then**
17: Map SFCRs into server *i*, *map sfcrs*().
18: Establish routing paths of flows using the shortest path algorithm.
19: **else**
20: Break.
21: **end**
22: **end**
23: **if** Γ *is not empty* **then**
24: Pick up the SFCR that demands the least resource, and try to map it into server *i*, *map last sfcr*().
25: Reserve redundancy for SFCRs in server *i*, *reserve redundancy*().
26: Establish routing paths of flows using the shortest path algorithm.
27: **else**
28: Break.
29: **end**
30:**end**

Algorithm 3: Map SFCRS

1:**Input**: Server *i*, Γ
2:**Output**: Mapping results of Γ to server *i*
3:**while** *SFCR with the least resource demand can be hosted on server i* **do**
4: **for** γ *in* Γ **do**
5: **if** *server i can hold SFCR* γ **then**
6: **if** *the first SFCR to be mapped* **then**
7: Map SFCR γ into server *i*.
8: **else**
9: **if** *is_relevant(SFCR* γ*, server i)* **then**
10: Map SFCR γ into server *i*.
11: **else**
12: Continue.
13: **end**
14: **end**
15: Remove SFCR γ from Γ.
16: **else**
17: Continue.
18: **end**
19: **end**
20: Relax the relativity requirement.
21:**end**

iterative process settles down which VNFRs should be hosted on the same multi-tenant capable VNF instance, and then the DRC demand of the VNF instance is determined by the DRC demands of VNFRs on it subsequently. The above processes solve **Q2** and the last part of **Q3**.

Nextly, JDBS is described in detail.

B. Availability Aware VNF Deployment

Algorithm 2 (JDBS) along with Algorithm 3, Algorithm 4, Algorithm 5 and Algorithm 6 shows the details of JDBS.

JDBS consists of a series of iterative processes (lines 5-30 in Algorithm 2). Firstly, all SFCRs are sorted in descending order based on their resource demands (line 3 in Algorithm 2), and then they are mapped into servers. During the process, each SFCR is mapped into a server (server *i*) as a whole (lines 7 and 17 in Algorithm 2), which means that all VNFRs in the SFCR are mapped into the same server. In this way, the flows between VNFRs are restricted in the same server, raising as less bandwidth consumptions as possible.

Based on above mapping process, the types of VNFRs in one server can be very various. However, only VNFRs of the same type can be hosted on the same multi-tenant capable VNF. So the more types of VNFRs there are, the more VNFs need to be instantiated in the server, then more BRC is required as a result. To reduce the types of VNFRs in a server, the relativity is considered as the measure to decide which SFCR should be mapped into the server preferentially (line 9 in Algorithm 3). In the process, the difference between the types of VNFRs in the SFCR to be mapped and the types of VNFRs already in the server is used as the measure of relativity. It is more relevant if the difference is smaller, and we map the most relevant SFCR firstly. In this way, there are more copies in a server for the same type of VNFR, so that multi-tenancy technology can be utilized more adequately, leading to less VNF instances. Then BRC can be reduced. To reduce the types of VNFRs in a server as much as possible, we relax the measure of relativity stepwise (line 20 in Algorithm 3), until the SFCR with the least resource demand cannot be hosted on server *i*.

When the SFCR with least resource demand cannot be mapped into server *i*, we stop the mapping process and start to deploy related VNFs. Initially, VNFs are implemented in single-tenancy principle, which means that each VNFR is corresponding to a VNF instance. If the availability requirement of any SFCR is not satisfied, redundancy needs to

Algorithm 4: reserve redundancy

1:**Input**: Server i, server $i+1$
2:**Output**: Server i, server $i+1$
3:Redundancy, $\varphi_1 = 0$.
4:Modify the availabilities of VNFs in server i.
5:**if** *all SFCRs' availability requirements are satisfied.* **then**
6: | **return** server i, server $i+1$.
7:**else**
8: | Find the VNF θ with the least DRC demand in server i, and its DRC is indicated as θ_r.
9: | **while** *Not all SFCRs' availability requirements are satisfied* **do**
10: | | $\varphi_1 = \varphi_1 + \theta_r$, modify the availabilities of all VNFs in server i, *modify availability*().
11: | **end**
12:**end**

Algorithm 5: Merge VNFRS

1:**Input**: Server *i*, server $i+1$
2:**Output**: Server *i*, server $i+1$
3:$saved\_brc = 0$.
4:**while** *there are VNFRs that can be merged* **do**
5: | $saved\_brc = saved\_brc + res^{brc}$.
6: | $res\_high = saved\_brc, res\_low = 0$.
7: | **while** $res\_high - res\_low > \delta$ **do**
8: | | Redundancy increment, $\varphi_2 = (res\_high + res\_low)/2$, modify the availability of all VNFs in server *i*, *modify availability*().
9: | | **if** *All SFCRs' availability requirements are satisfied* **then**
10: | | |
11: | | **else**
12: | | | $res\_high = \varphi_2$.
13: | | **end**
14: | | $res\_low = \varphi_2$.
15: | **end**
16: | $saved\_res = saved\_brc - \varphi_2$.
17: | **if** $saved\_res > 0$ *and* $saved\_res$ *is smaller than the last* **then**
18: | | Break.
19: | **else**
20: | | Continue.
21: | **end**
22:**end**

Algorithm 6: Map Last SFCR

1:**Input**: Last SFCR γ_l, server *i*, *P*
2:**Output**: Server *i*, *P*
3:**for** *VNFR in SFCR* γ_l **do**
4: | **if** *Server i can hold the VNFR* **then**
5: | | Map the VNFR into server *i*.
6: | **else**
7: | | Break.
8: | **end**
9:**end**
10:Map the rest VNFRs into server $i+1$ or other servers in *P* as a whole.

be reserved in the nearby server for the VNFs in server *i* (line 9 in Algorithm 2). To figure out the volume of needed redundancy while keeping resource utilization efficiency, the increment of redundancy in each iterative loop equals to the least DRC demand of all VNFs in the server (lines 8 and 10 in Algorithm 4).

After the backup process, the availability requirements of all SFCRs in server *i* are met. However, the number of VNFs is high owing to the single-tenant implementation of them, so the volume of BRC is high. As stated before, the number of VNFs can be reduced utilizing multi-tenancy technology. So an algorithm is designed to merge together the VNFRs of the same type (Algorithm 5). When merging two VNFRs together, one share of BRC will be saved (res^{brc}, line 5 in Algorithm 5). The VNFRs that are going to be merged together are chosen randomly. After that, VNFs need to be re-instantiated in multi-tenancy principle, and the availabilities of all VNFs change owing to the redundancy sharing mechanism. Moreover, the shared DRC redundancy may need to increase to improve the availabilities of VNFs in server *i*.

The binary search method is used to find out the increment of shared DRC redundancy (lines 7-14 in Algorithm 5). Then the difference between increment of shared DRC redundancy and saved BRC is calculated (line 15 in Algorithm 5) to figure out the final saved resource. If saved resource is less than before, we think that saved resource reaches maximum value, and then the merging process of VNFRs is stopped (lines 16 and 17 in Algorithm 5).

After the merging process, free resources will arise in server *i*, so new SFCRs can be mapped into the server again to make the best use of the network resource (line 17 in Algorithm 2). Then the merging process (Algorithm 5) also should be done again because of the new added VNFRs. Therefore, Algorithm 5 and Algorithm 3 form a loop (lines 10-22 in Algorithm 2). If any of the two processes failed, the loop is broken.

At last, server *i* cannot host any SFCR as a whole. Nevertheless, to make the best use of resources in server *i*, we can pick out the SFCR that has the least resource demand, and map part of the SFCR into server *i* (Algorithm 6). In Algorithm 6, VNFRs in the SFCR are mapped into the server one by one in sequence, until one VNFR cannot be hosted on the server. Then the rest of the SFCR is mapped into the nearby server. In this way, VNFRs in each of the separated part are kept in order, which can incur as less extra bandwidth consumption as possible.

C. Complexity

To figure out the time complexity of JDBS, we must figure out the time complexity of all included algorithms.

For Algorithm 1, the time complexity is at the level of $\mathbf{O}(\mathbb{N} \cdot 2^{\mathbb{N}})$ because of Cartesian power. $\mathbb{N} = C/(\omega_m + brc_u)$ at most, C indicates the CPU or memory capacity of the server, ω_m indicates the minimal CPU or memory DRC of all VNFRs and brc_u indicates BRC when instantiating a VNF instance.

For Algorithm 3, the difference of VNFR types is $|\Phi|$ at most, so the *while* loop runs $|\Phi|$ times at most, then the complexity of Algorithm 3 is at the level of $\mathbf{O}(|\Phi| \cdot |\Gamma|)$. For Algorithm 4, the main complexity relies on the *while* loop (lines 9-17 in Algorithm 4), and its iteration times is $\mathbb{N}$ at most. So the time complexity of Algorithm 4 is at the level of $\mathbf{O}(\mathbb{N}^2 \cdot 2^{\mathbb{N}})$.

Then for Algorithm 5, the binary search method is used to find out the needed redundancy increment, and the time complexity of the binary search method is $\mathbf{O}(log_2 C)$ at most. Then the time complexity of Algorithm 5 is at the level of $\mathbf{O}(\mathbb{N} \cdot log_2 C \cdot \mathbb{N} \cdot 2^{\mathbb{N}})$. The time complexity of Algorithm 6 is at the level of $\mathbf{O}(1)$. In summary, the dominate time complexity relies on Algorithm 5, which is at the level of $\mathbf{O}(\mathbb{N}^2 \cdot 2^{\mathbb{N}} \cdot log_2 C)$.

Finally, for lines 8, 18 and 26 in Algorithm 2, the time complexity of the shortest path algorithm is at the level of $\mathbf{O}(|N^s| \cdot log_2|N^s|)$, $|N^s|$ indicates the number of substrate nodes in the network. We need to establish a path for each SFCR, so the time complexity of establishing paths is $|\Gamma| \cdot |N^s| \cdot log_2|N^s|$. Furthermore, the time complexity of Algorithm 2 relies on the complexity of Algorithm 3, Algorithm 4 and Algorithm 5 and the iterative times of the two *while* loops in Algorithm 2 is at the level of $\mathbf{O}(|\Gamma|^2)$. However, because $log_2 C \cdot \mathbb{N}^2 \cdot 2^{\mathbb{N}} \gg \mathbb{N}^2 \cdot 2^{\mathbb{N}}$. So the total time complexity of JDBS is at the level of $\mathbf{O}(|\Phi| \cdot |\Gamma|^3 + |\Gamma|^2 \cdot \mathbb{N}^2 \cdot 2^{\mathbb{N}} \cdot log_2 C + |\Gamma| \cdot |N^s| \cdot log_2|N^s|)$, $\mathbb{N} = C/(\omega_m + brc_u)$, which relies on the number of SFCRs, types of VNFs, the ratio between the capacities of servers and resource demands of VNFRs, and the scale of substrate network.

V. Numerical Simulation

In this section, the performance of JDBS is compared with 4 contrasting schemes and an analysis about the results is made in detail. JDBS and the contrasting schemes are implemented in Python. All experiments are performed on a computer with one Intel(R) Core(TM) i7-6700 CPU @ 2.6GHz and 16GB of RAM.

A. Simulation Settings

We use the fat-tree topology as the structure of datacenter [42]. The number of servers in the substrate network is set to be 1024. For each server, it has 1000 units available CPU resource and 1000 units available memory resource. The available bandwidth of each link is also set to be 1000 units. Both CPU BRC and memory BRC when instantiating a VNF are set to be 20 units.

For SFCRs, the number of VNFRs in each SFCR is a random integer value from [3,4,5,6]. The CPU DRC, memory DRC and bandwidth demand of each VNFR all obey uniform distribution of (10, 50) units. There are 10 types of VNFs in the simulation. Moreover, both the inherent availability of primary VNFs and the availability of backup VNFs obey the uniform distribution of (0.99, 0.999). The number of SFCRs and the availability requirements of SFCRs are treated as variables.

$\delta = 1$ in Algorithm 5.

B. Benchmarks

To validate the performance of JDBS, the following contrasting schemes from existing literatures are used:

1) *RAR_RS_BiG:* RAR_RS_BiG is the solution proposed in [6]. According to RAR_RS_BiG, each SFC is augmented by adding backup VNFs to it firstly, which aims to have the availability requirement of each SFC satisfied. When reserving backup VNFs, the backup VNF can be shared by the adjacent VNFs to improve resource efficiency. Then the SFCs are mapped into the network using Bi-direction search based greedy shortest path algorithm.
2) *JDBS_s:* JDBS_s is the variant of JDBS. The difference is that JDBS_s implements VNFs in single-tenancy principle.
3) *GREP:* GREP is the solution proposed in [8], which improves the availability of an SFC by providing backup for the most unreliable VNFs.
4) *CERA:* CERA is the solution proposed in [29], which claims to be the modification of GREP. The authors in [29] found that providing backups for different VNFs in an SFC gets different availability improvements with different costs. So the importance of each VNF in an SFC is different. In order to improve resource efficiency, it selects the VNF that has the largest cost-aware importance measure (CIM) to backup, in which CIM is the ratio between availability improvement of the SFC and the cost to pay when providing backup for a VNF.

All contrasting schemes implement the VNF instances in single-tenancy principle.

C. Results

The simulation results are shown in this part. For each group of results, 10 times of experiments are conducted to reduce the accidental errors, and error bars represent the 95% confidence intervals in each figure.

In the simulations, it is assumed that all SFCRs can be settled down on the substrate network. So in each group of results, the number of used servers, CPU and memory DRC redundancy, bandwidth consumptions and final BRC are used as the measures to evaluate the performance of different solutions. It is worth noting that BRC is classified into primary BRC, which is corresponding to primary VNFs, and backup BRC, which is corresponding to backup VNFs.

1) Comparisons With Varying Availability Requirements of SFCRs: Fig. 2, Fig. 3, and Fig. 4 show the number of used servers, reserved DRC redundancy and BRC respectively by different solutions along with varying availability requirements of SFCRs. In this scenario, there are 1000 SFCRs in each group of experiment.

As Fig. 2 shows, JDBS costs the least servers in each group of experiment, which can save 29%, 29%, 42%, 40%

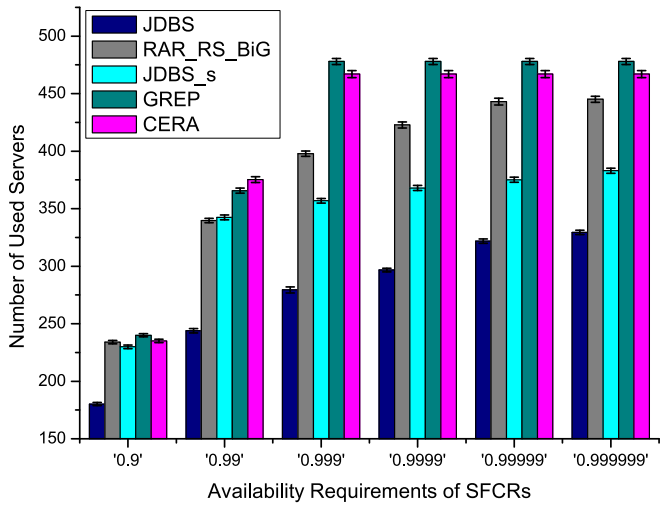


Fig. 2. Number of used servers vs varying availability requirements of SFCRs.

servers at most than RAR_RS_BiG, JDBS_s, GREP, CERA, respectively.

As stated before, JDBS_s is the single-tenancy version of JDBS, which also takes advantage of the same deployment scheme and redundancy sharing mechanism as JDBS. Comparing JDBS_s with RAR_RS_BiG, we can see that they have similar performance when the availability requirements of SFCRs are lower than the inherent availabilities of VNFs (scenarios that availability requirements are 0.90 and 0.99). It is because the availability requirements of SFCRs can be satisfied by current VNFs, and there is no need to reserve redundancy to improve the availabilities of VNFs.

In Fig. 3(b) and Fig. 3(c), we can see that both CPU DRC redundancy and memory DRC redundancy are 0 in results of all 5 solutions, when the availability requirement is 0.90. Also as is shown in Fig. 4(c), the backup BRC of all 5 solutions are 0 at the point of 0.90. The above phenomenons indicate that there is no need to reserve redundancy to improve the availabilities of VNF instances when the availability requirement of SFCRs is 0.90.

When the availability requirements of SFCRs increase to 0.99, they cannot be satisfied by current VNFs, and redundancy needs to be reserved to improve the availability of each VNF. However, the requirement is not so high, and a small quantity of redundancy can improve the availability to a qualified level. From Fig. 3(b) and Fig. 3(c), it can be seen that CPU DRC redundancy and memory DRC redundancy by both JDBS and JDBS_s are lower than that of RAR_RS_BiG. It is because DRC redundancy in the server is shared by all VNFs in another server, rather than shared by the adjacent VNFs belonging to the same SFC in RAR_RS_BiG. JDBS and JDBS_s can acquire a better sharing efficiency than RAR_RS_BiG.

In Fig. 2, we can see that the number of used servers by JDBS_s is a little higher than that of RAR_RS_BiG when availability requirement is 0.99. It is because JDBS_s has to reserve BRC for all VNFs to provide the active-standby mode backup scheme. So the backup BRC by JDBS_s is much higher than that of RAR_RS_BiG, which can be seen in Fig. 4(c).

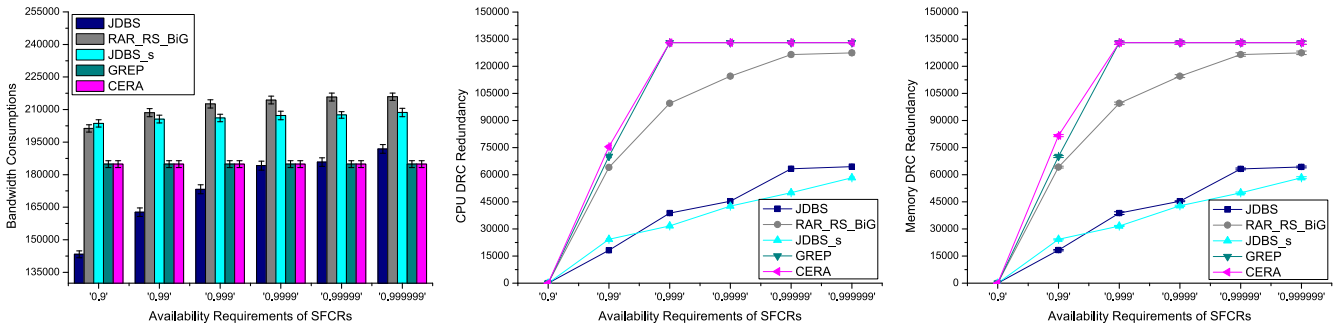
With the increasing of availability requirement, CPU redundancy, memory redundancy, and BRC redundancy by RAR_RS_BiG all increase. Meanwhile, the surplus of CPU and memory redundancy between RAR_RS_BiG and JDBS_s is more, as are shown in Fig. 3(b) and Fig. 3(c). The deficit of total BRC between RAR_RS_BiG and JDBS_s is less, as is shown in Fig. 4(a). So the number of used servers by RAR_RS_BiG is getting more and more than that of JDBS_s.

As for GREP and CERA, they share similar deployment and backup philosophy, except that CERA takes different importance of VNFs into consideration, to improve the redundancy efficiency. Fig. 2 shows that CERA can acquire a lower number of used servers than that of GREP. However, both of them do not utilize the redundancy sharing mechanism to improve the redundancy efficiency. When the availability requirement is higher than a threshold, which is 0.999 in the simulation, both of them have to reserve redundancy for all VNFs. Therefore, both resource consumptions and BRC reach maximum values, which can be seen in Fig. 3 and Fig. 4.

Fig. 3(a) shows the bandwidth consumptions by 5 solutions. From the figure, we can see that RAR_RS_BiG consumes the most bandwidth, and bandwidth consumptions by CERA and GREP are close to each other. Bandwidth consumptions by JDBS increase with the increasing of availability requirement, which results from the increasing number of used servers and the separating of SFCRs. In JDBS, to improve the resource utilization efficiency, every server is filled with SFCRs as full as possible. If one server cannot host any SFCR wholly, the SFCR that consumes the least resource is departed into two parts to be mapped into the server and its neighbor. So extra bandwidth consumptions occur between servers, and more servers result in more bandwidth consumptions in JDBS.

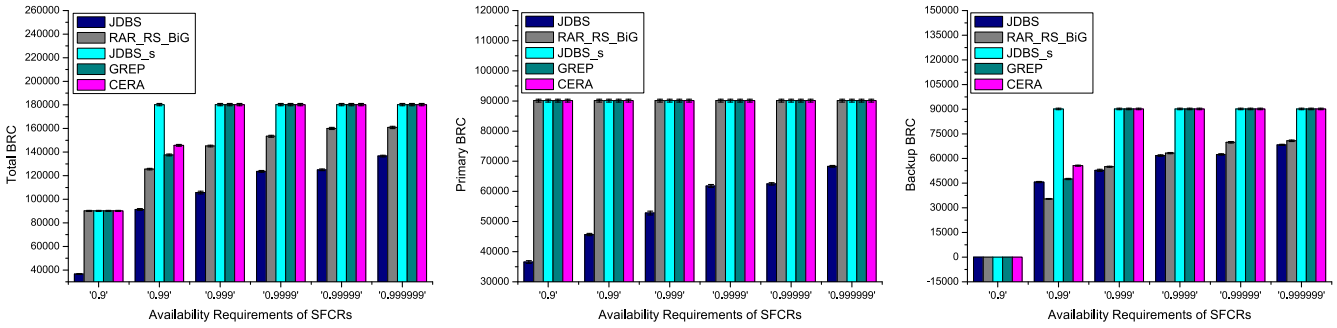
In most scenarios, bandwidth consumptions by JDBS are lower than that of contrasting schemes, owing to multi-tenancy technology. Because multi-tenancy are utilized to reduce the number of VNFs, then more SFCRs are restricted into one server than that of contrasting schemes, leading to a much lower number of used servers. For RAR_RS_BiG and JDBS_s, both of them result in extra bandwidth between servers, therefore, their bandwidth consumptions are higher than JDBS because of more used servers. For CERA and GREP, when an SFCR cannot be mapped into one server, both of them will not make further processes, so there are no extra bandwidth consumptions between servers. Moreover, flows inner the SFCRs are restricted into servers, and bandwidth consumptions occur along the links between the ingress/egress of the datacenter and the servers for SFCRs. Thus the bandwidth consumptions are fixed for the same set of SFCRs.

From Fig. 4, we can see that primary BRC is fixed for RAR_RS_BiG, JDBS_s, GREP and CERA. The reason is that all of these algorithms implement VNFs in single-tenancy principle for the same set of VNFRs, so primary BRC of them is the same. As for JDBS, which utilizes multi-tenancy to implement VNF instances, the primary BRC increases with the increasing of availability requirements. The reason is that VNFRs on each multi-tenant capable VNF tends to be less



(a) Bandwidth consumptions (b) CPU DRC redundancy (c) Memory DRC redundancy

Fig. 3. Reserved redundancy vs varying availability requirements of SFCRs.



(a) Total BRC (b) Primary BRC (c) Backup BRC

Fig. 4. BRC vs varying availability requirements of SFCRs.

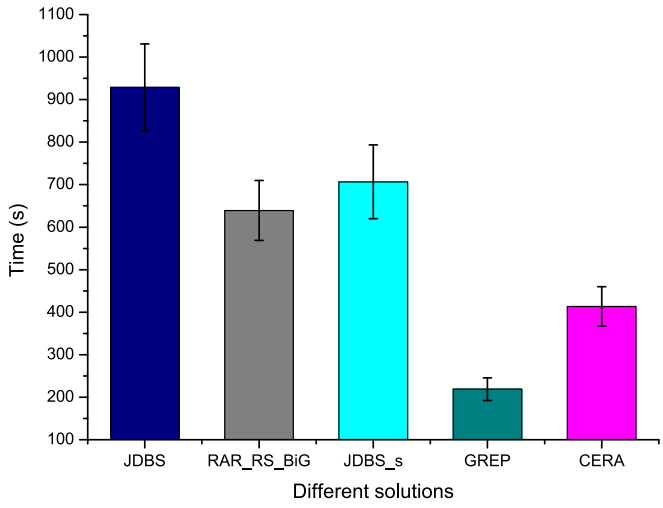


Fig. 5. Time required to execute one solution.

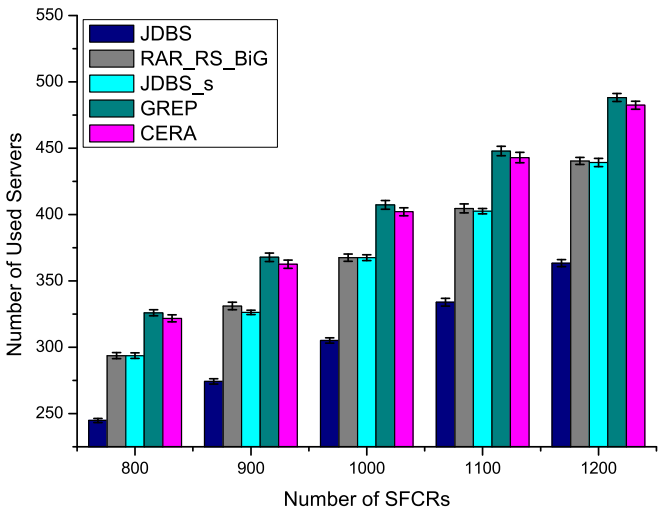
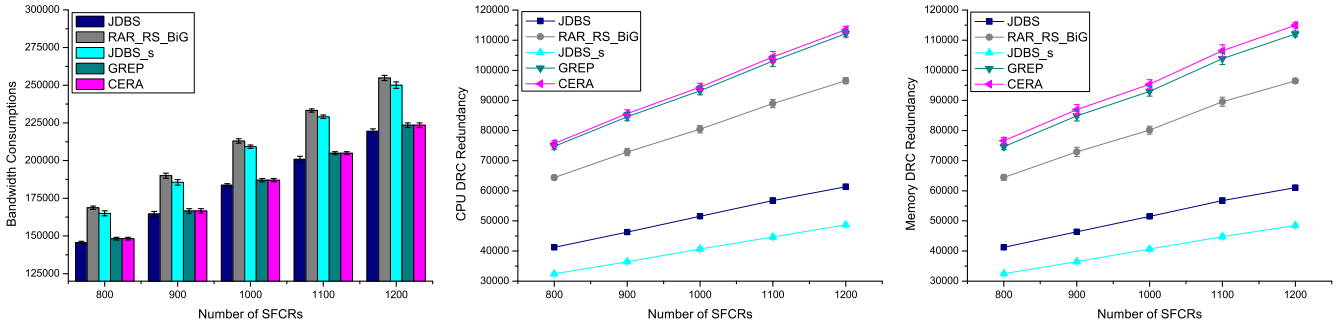


Fig. 6. Number of used servers vs varying SFCR number.

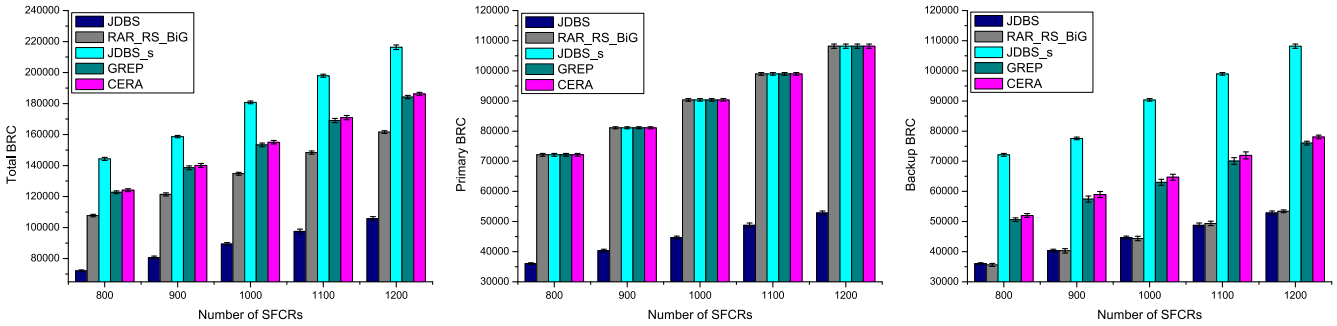
and the number of VNFs becomes larger with the increasing of availability requirements, in order to make a balance between BRC and increasing shared redundancy. Generally, backup BRC by JDBS is the same as primary BRC, because we reserve redundancy for each primary VNF. However, when the availability requirement is 0.90, there is no need to reserve redundancy for primary VNFs, so the backup BRC is 0 in this scenario. For GREP and CERA, it can be seen that their backup BRC reaches the maximum value when the availability requirement is higher than 0.999, which indicates that GREP and CERA reserve redundancy for all primary VNFs in these scenarios.

Fig. 5 shows the comparisons of time required to execute one instance of different solutions when the availability requirement is 0.99999. From the figure, we can see that JDBS consumes the most time. Based on detailed analysis, it can be found that the most time-consuming part of JDBS is Algorithm 5. Because it involves multiple times of availability modification calculating, which is in exponential complexity. However, the time complexity is constrained, where the



(a) Bandwidth consumptions (b) CPU DRC redundancy (c) Memory DRC redundancy

Fig. 7. Reserved redundancy vs varying SFCR number.



(a) Total BRC (b) Primary BRC (c) Backup BRC

Fig. 8. BRC vs varying SFCR number.

exponent is the ratio between the capacity of server and the minimal resource demand of the VNFRs.

2) Comparisons With Varying SFCR Number: Fig. 6, Fig. 7, and Fig. 8 show the number of used servers, reserved DRC redundancy and BRC respectively by different solutions with varying number of SFCRs. In this scenario, the availability requirements of SFCRs are chosen from [0.90, 0, 99, 0.999, 0.9999, 0.99999, 0.999999] randomly.

In general, consistent conclusions can be derived with the results based on different availability requirements of SFCRs. From the figures, we can see that JDBS costs the least number of servers and the least total BRC for the same set of SFCRs, which can save 18%, 18%, 26%, 25% servers at most than RAR_RS_BiG, JDBS_s, GREP, CERA, respectively.

It is worth noting that JDBS consumes more CPU and memory DRC redundancy than that of JDBS_s, which can be seen in Fig. 7(b) and Fig. 7(c). However, BRC by JDBS is much lower than that of JDBS_s, as is shown in Fig. 8. It strengthens the forward conclusion that smaller volume of VNFs can acquire better efficiency in the redundancy sharing mechanism, which is that they need less shared redundancy. Nevertheless, the number of VNFs is more, and so is BRC.

3) Comparisons With Optimal Results: We reveal the gap between the performance of JDBS and the optimal results in this part. A brute force solution is used to get the optimal results in a small scale scenario, where there are 100 SFCRs to be mapped and the availability requirements of all SFCRs are 0.99999. Fig. 9 shows the performance comparisons of different solutions with the optimal results. From the figure, we can see that JDBS can acquire a near-optimal performance.

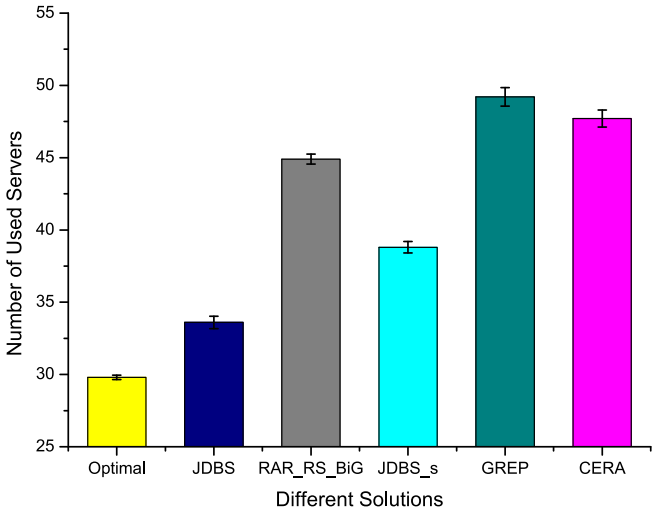


Fig. 9. Performance comparisons of different solutions with optimal results.

VI. Conclusion

In this paper, we study the availability aware VNF deployment problem given a set of SFCRs. To improve the resource utilization efficiency, the redundancy sharing mechanism and multi-tenancy technology are considered. Furthermore, we get the conclusion that VNFs with smaller volume need less shared DRC redundancy to get the same availability improvement. But BRC is more as a result, owing to more VNFs. So there is a tradeoff between BRC and shared DRC redundancy.

The availability aware VNF deployment problem is formulated mathematically, and a joint VNF deployment and backup solution, JDBS, is proposed. In JDBS, an availability modification process is designed to figure out the modified availability of each VNF based on a given block of shared redundancy. Moreover, a merging process is proposed to figure out which VNFRs should be hosted the same multi-tenant capable VNF. Finally, the performance of JDBS is evaluated through numerical simulations in detail with 4 solutions in existing literatures, and the simulation results show that JDBS outperforms the contrasting schemes and can save about 40% resources than them at most.

VII. Discussion

NFV is a promising and evolving technology to enhance the next generation network, such as 5G [43], next generation datacenters [44], multi-access edge computing (MEC) [45]. Also VNF deployment problem has attracted lots of attention from researchers. VNFs are software-based network components, and the nature and technology development of software should be considered in the VNF deployment problem, to improve the QoS of service and resource utilization efficiency. However, there are not many related researches now.

In our paper, we consider the error prone nature of VNFs, and aim to design an availability aware VNF deployment solution. Moreover, the virtualization overheads are considered when instantiating VNFs, which is BRC in our model. We deal with the virtualization overheads and resource needed to accomplish the service separately, and design an active-standby backup scheme based on the on-demand resource allocation mode of NFV-MANO. Besides, we find that redundancy sharing mechanism and multi-tenancy technology can be utilized properly to improve the resource efficiency.

During these formulations, we found that it is hard to express mathematically the availability aware problem based on shared redundancy. Therefore further research is needed to make suitable optimizations. In addition, our ideas about virtualization overheads need more theories and experimentation to validate them.

Acknowledgment

The authors sincerely thank the editor, Dr. David Hutchison, and all the anonymous reviewers for their valuable suggestions that have led to the present improved version of the original manuscript.

References

[1] 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.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[3] G. Gibb, H. Zeng, and N. McKeown, "Outsourcing network functionality," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 73–78.

[4] 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," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, 2012.

[5] J. Kang, O. Simeone, and J. Kang, "On the trade-off between computational load and reliability for network function virtualization," *IEEE Commun. Lett.*, vol. 21, no. 8, pp. 1767–1770, Aug. 2017.

[6] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, Mar. 2018.

[7] Industry Specification Group, *Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability, V1*, document GS NFV-REL 003, ETSI, Sophia Antipolis, France, 2016.

[8] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Funct. Virtualization*, 2015, pp. 13–18.

[9] L. Chen, S. Patel, H. Shen, and Z. Zhou, "Profiling and understanding virtualization overhead in cloud," in *Proc. IEEE ICPP*, 2015, pp. 31–40.

[10] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[11] *Containers vs VMs Technology Smackdown*, Bright Comput., San Jose, CA, USA, May 2016. [Online]. Available: http://www.brightcomputing.com/blog/containerization-vs.-virtualization-heres-our-blog-smackdown

[12] P. C. Rangarajan, F. Khendek, and M. Toeroe, "Managing the availability of VNFS with the availability management framework," in *Proc. IEEE CNSM*, 2017, pp. 1–4.

[13] Industry Specification Group, *Network Function Virtualisation (NFV)-Resiliency Requirements*, document GS NFV-REL 001, ETSI, Sophia Antipolis, France, 2014.

[14] F. Carpio and A. Jukan, "Improving reliability of service function chains with combined VNF migrations and replications," *arXiv preprint arXiv:1711.08965*, 2017.

[15] M. Ersue, "ETSI NFV management and orchestration-An overview," in *Proc. 88th IETF Meeting*, 2013.

[16] Z. Xu, W. Liang, A. Galis, Y. Ma, Q. Xia, and W. Xu, "Throughput optimization for admitting NFV-enabled requests in cloud networks," *Comput. Netw.*, vol. 143, pp. 15–29, Oct. 2018.

[17] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE IM*, Ottawa, ON, Canada, 2015, pp. 98–106.

[18] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[19] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. CloudNet*, 2014, pp. 7–13.

[20] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *IEEE J. Lightw. Technol.*, vol. 33, no. 8, pp. 1565–1570, Apr. 15, 2015.

[21] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. INFOCOM*, Hong Kong, 2015, pp. 1346–1354.

[22] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 343–356, Jun. 2017.

[23] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Services Comput.*, to be published.

[24] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May/Jun. 2016.

[25] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Commun.*, vol. 23, no. 3, pp. 90–96, Jun. 2016.

[26] M. Di Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco, "Service function chaining deployed in an NFV environment: An availability modeling," in *Proc. IEEE CSCN*, 2017, pp. 42–47.

[27] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5G mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 483–496, Mar. 2016.

[28] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Proc. IEEE IWQoS*, 2017, pp. 1–10.
[29] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme," in *Proc. IEEE ICC*, 2017, pp. 1–6.
[30] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, Oct. 2017.
[31] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *Proc. IEEE NFV-SDN*, Palo Alto, CA, USA, 2016, pp. 128–133.
[32] L. Qu, C. Assi, K. Shaban, and M. 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.
[33] C.-P. Bezemer and A. Zaidman, "Multi-tenant SaaS applications: Maintenance dream or nightmare?" in *Proc. EVOL IWPSE*, 2010, pp. 88–92.
[34] S. Pal, A. K. Mandal, and A. Sarkar, "Application multi-tenancy for software as a service," *SIGSOFT Softw. Eng. Notes*, vol. 40, no. 2, pp. 1–8, 2015.
[35] A. M. Medhat, G. Carella, J. Mwangama, and N. Ventura, "Multi-tenancy for virtualized network functions," in *Proc. IEEE NetSoft*, 2015, pp. 1–6.
[36] Z. Á. Mann and A. Metzger, "Optimized cloud deployment of multi-tenant software considering data protection concerns," in *Proc. CCGrid*, 2017, pp. 609–618.
[37] Y. Zhang, Z. Wang, B. Gao, C. Guo, W. Sun, and X. Li, "An effective heuristic for on-line tenant placement problem in SaaS," in *Proc. IEEE ICWS*, 2010, pp. 425–432.
[38] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *Proc. IEEE ICC*, 2016, pp. 1–6.
[39] R. Yu, G. Xue, X. Zhang, and D. Li, "Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers," in *Proc. INFOCOM*, Atlanta, GA, USA, 2017, pp. 1–9.
[40] J. Lee *et al.*, "Application-driven bandwidth guarantees in datacenters," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 467–478, 2014.
[41] *Cartesian Product*, Wikipedia, San Francisco, CA, USA, Jun. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Cartesian_product
[42] R. Niranjan Mysore *et al.*, "Portland: A scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.
[43] F. van Lingen *et al.*, "The unavoidable convergence of NFV, 5G, and fog: A model-driven approach to bridge cloud and edge," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 28–35, Aug. 2017.
[44] A. Sheoran, P. Sharma, S. Fahmy, and V. Saxena, "Contained: An NFV micro-service system for containing E2E latency," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 5, pp. 54–60, 2017.
[45] B. Blanco *et al.*, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC, and SDN," *Comput. Stand. Interfaces*, vol. 54, pp. 216–228, Nov. 2017.

Defang Li received the B.S. degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China in 2014, where he is currently pursuing the Ph.D. degree. His research interests include SDN, NFV, and the network resource orchestration and management.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China in 1983 and 1986, respectively, where she is currently a Professor and the Advisor for Ph.D. candidates. She has published 2 books and over 100 academic papers in several journals and conference proceedings. Her research interests include next-generation Internet, policy control, IP QoS, and information security.

Kaiping Xue (M'09–SM'15) received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC in 2007, where he is currently an Associate Professor. His research interests include next-generation Internet, distributed networks, and network security. He currently serves as an Area Editor for *Ad Hoc Networks* and an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE ACCESS, and *China Communications*. He also served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and a Lead Guest Editor for the *IEEE Communications Magazine*. He is an IET Fellow.

Jianing Pei received the B.S. degree from the Department of Information and Electrical Engineering, China University of Mining and Technology in 2015. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include SDN, NFV, and the network resource orchestration and management.