Availability Aware VNF Deployment in Datacenter Through Shared Redundancy

Defang Li, Peilin Hong, Jianing Pei, Wenzhe Wang
The Key Laboratory of Wireless-Optical Communications, Chinese Academy of Sciences,
School of Information Science and Technology, University of Science and Technology of China, Hefei, 230027, China.
ldf911@mail.ustc.edu.cn, plhong@ustc.edu.cn, jianingp@mail.ustc.edu.cn, wzwang@mail.ustc.edu.cn

***Abstract*—Network function virtualization (NFV) has brought great cost reducing and operation flexibility to network services, in which users' service requests are accomplished by softwares on the common-off-shelf servers rather than dedicated proprietary hardware middleboxes. Then how to guarantee the availability of these services is coming correspondingly owing to the error prone nature of softwares. Resource redundancy has been seen as an efficient way. Moreover, the resource orchestration of softwares is more flexible than that of physical machines. Therefore, how to design a resource efficient solution to assure the availability of network services in NFV environment has been attracting attentions in academics and industries. Network services are usually finished by service function chains (SFC) in NFV, and an SFC is composed by several virtual network functions (VNF) in order. In this paper, we study the availability aware VNF deployment problem considering users' SFC requests (SFCr). To improve the resource efficiency, a Joint Deployment and Backup scheme taking advantage of Shared Redundancy (JDBSR) is proposed. Through the thorough simulations, the results show that our solution has a great advantage over the benchmarks.**

***Index Terms*—Network Function Virtualization, SFC, VNF Deployment, Availability, Shared Redundancy**

I. Introduction

Network function virtualization (NFV) enhances the flexibilities and conveniences of cloud services. With the help of NFV, many small and medium businesses can outsource their IT infrastructures to the cloud, saving a great of capital expenditures and operating expenses (CAPEX/OPEX) as a result. Generally, the network services (NS) proposed by users are accomplished by service function chains (SFCs) in NFV, which are composed of a series of virtual network functions (VNFs) in specified order.

Despite the advantages that NFV brings, how to deploy the related VNFs to serve the tenants in cloud optimally is a tricky problem, which is usually known as the VNF deployment/placement problem [1][2]. And researchers have made plenty of efforts with diverse optimal targets and constraints. The targets can be minimizing the total resource consumption [1], minimizing the number of consumed physical machines (PMs) [2], and minimizing the total service delay [3]. As for the constraints, resource capacity, service latency are two mostly considered factors.

In this paper, we study the VNF deployment problem considering availability of users' SFC requests (SFCr), which are different from each other. To meet these availability requirements, redundancy is the de-facto technology [4]. But a resource efficient redundancy scheme needs to be well designed.

In [4], the reliability/availability model and principle about how to assure the end to end (E2E) reliability/availability of SFCs are demonstrated clearly by European Telecommunications Standards Institute (ETSI). In [5], the authors tended to backup the most unreliable VNFs to improve the total reliability of SFCs. Ding et al. in [6] improved the selection model of VNFs to be backuped based on the Cost-aware Importance Measure (CIM) they designed, to improve the backup cost efficiency. Also, researchers in [7][8][9] proposed their solutions to make a balance between the backup cost and the reliability requirements of SFCs.

Redundancy is also used in our scheme to guarantee the availability. Compared with the works in existing literatures, the pattern of VNF resource consumption is classified more specifically in this paper, which is divided into two kinds. One is the overhead when instantiating a VNF, for example, the resource consumption to maintain the image of the VNF. It is called basic resource consumption (BRC) in our previous work [10]. The other part is computing resource to accomplish the regular functioning of one VNF, and the computing redundancy can be pooled and shared by different VNFs utilizing the resource scaling scheme [11][12], to improve the resource efficiency of the backup scheme.

In summary, given a set of SFCrs proposed by users, we need to map them into the cloud datacenter, and then deploy and chain the related VNFs to serve them, while guaranteeing availability requirement of each SFCr. To make a balance between availability and backup cost, the sharing mechanism of computing redundancy is used. The problem is NP-hard, so an efficient heuristic is proposed, named Joint Deployment and Backup scheme through Shared Redundancy(JDBSR). Our major contributions can be summarized as below:

- We clarify the overhead (BRC) and computing resource clearly when instantiating VNFs.
- We utilize the sharing mechanism to improve the redundancy efficiency when providing backups to the primary VNFs, and clarify the availability modification model based on the shared redundancy.
- We formulate the availability aware VNF deployment problem and propose a Joint Deployment and Backup scheme utilizing Shared Redundancy, JDBSR. Then the performance of the solution is evaluated through numeri-

cal simulations in detail, and the results show that JDBSR outperforms the benchmarks apparently.

The rest of our paper is organized as follows. We state and formulate the problem in Section II. Then the proposed solution is described in Section III, and Section IV demonstrates the simulation results. Finally, our work is concluded in Section V.

II. Problem Statement and Formulation

In this section, we make clear some related concepts firstly, and then demonstrate and formulate the problem respectively.

A. Network Model

Fat-tree [13] is the frequently used topology in datacenter [14][15]. In this topology, the size of the network is determined by the number of ports in the switches. For a k-ary fat-tree topology, there are $(k/2)^2$ k-port core switches; k pods, each of which containing two layers of $k/2$ switches; and $k^3/4$ PMs.

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

B. Availability Model For an SFC

Link and switch failures are not considered, as modern datacenters typically have rich path diversity between any pair of PMs, which can effectively protect over these failures [15][16]. So we only consider the VNF failures. Moreover, the failures among the VNFs are usually assumed as independent [5][6]. Generally, the availability of an SFC is:

$$\mathbb{P} = \prod_{i=0}^{m-1} p_i \tag{1}$$

where m is the total number of VNFs in the SFC, and p_i indicates the availability of VNF i in the SFC respectively.

The backup VNFs are used to take place of the failed primary VNFs temporarily, and the services need to be redirected to the restored primary VNFs [4]. So it is assumed that the backup VNFs will not be outage when they are in service.

C. Problem Statement

Given a set of SFCrs, we need to design a solution to map these SFCrs into the datacenter network, and then deploy the related VNFs and chain them together to provide service to these SFCrs, while guaranteeing the availability requirement of each SFCr utilizing backup VNFs. In the problem, the following three questions need to be solved:

- **Q1:** How to coordinate the relationship between VNF deployment and backup ?
- **Q2:** Where to place these backup VNFs ?
- **Q3:** How many resources should be allocated to each backup VNF ?

For a given set of SFCrs, the single-tenancy principle is applied to the implementation of primary VNFs for the sake of performance isolation. So each VNFr is corresponding to a VNF instance in the substrate network, and each SFCr is corresponding to an SFC.

D. Problem Formulation

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

TABLE I: Notations

Pramaters	Descriptions
SFC related	
Γ	set of total SFCs, $\gamma \in \Gamma$ is an SFC.
Ψ_γ	set of total VNFs in SFC γ.
E^ν_γ	set of logical links between the nodes of SFC γ.
n^ν_i	one node in an SFC.
(n^ν_i, n^ν_j)	logical link between n^ν_i and n^ν_j.
$p_0(n^\nu_i)$	inherent availability of VNF n^ν_i.
$p_\varepsilon(n^\nu_i \leftarrow n^s_u)$	modified availability of VNF n^ν_i whose backup is on PM n^s_u.
A_γ	availability requirement of SFC γ.
Topology related	
N^s	set of total nodes in substrate network.
P	set of total PMs.
E^s	link set of the substrate network.
n^s_u	one node 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 consumption by VNF n^ν_i in SFC γ.
$b_{\gamma,n^\nu_i,n^\nu_j}$	bandwidth consumption by logical link $(n^\nu_i\ n^\nu_j)$ in SFC γ.
$\mathrm{cpu}^{\mathrm{brc}}_{\gamma,n^\nu_i}$	CPU BRC when instantiating an instance of VNF n^ν_i in SFC γ.
$C^{\mathrm{cpu}}_{n^s_u}$	CPU capacity of PM n^s_u.
$C^{\mathrm{link}}_{(n^s_u,n^s_v)}$	link capacity of substrate link (n^s_u, n^s_v).
$R^{\mathrm{bk}}_{\mathrm{comp}}(n^s_u)$	shared computing resource redundancy in PM n^s_u for SFCs on other PMs.
Binary variables	
x_{γ,n^ν_i,n^s_u}	whether n^ν_i in SFC γ is hosted on substrate node n^s_u, if yes, x_{γ,n^ν_i,n^s_u} = 1, $n^s_u \in P \cup R$.
z_{γ,n^ν_i,n^s_u}	whether the backup of VNF n^ν_i in SFC γ is on PM n^s_u, if yes, z_{γ,n^ν_i,n^s_u} = 1.
$y_{\gamma,n^\nu_i,n^\nu_j,n^s_u,n^s_v}$	whether the logical link (n^ν_i, n^ν_j) in SFC γ is mapped on 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.
$h_{n^s_u}$	whether PM n^s_u is used.

Considering the tradeoff between bandwidth optimization and node resource optimization, the optimization target is set as minimizing the number of used PMs:

$$\min \sum_{u=0}^{|P|-1} h_{n^s_u} \tag{2a}$$

$$h_{n^s_u} = \begin{cases} 1, & \sum_{\gamma=0}^{|\Gamma|-1} \sum_{i=0}^{|\Psi_\gamma|-1} x_{\gamma,n^\nu_i,n^s_u} \geq 1; \\ 0, & \text{otherwise.} \end{cases} \tag{2b}$$

Eq. 2b indicates that if more than one VNF is hosted on PM n^s_u, then the PM has to be activated.

Nextly, the node resource constraints are introduced:

$$R^{\text{pri}}_{\text{comp}}(n^s_u) = \sum_{\gamma=0}^{|\Gamma|-1} \sum_{i=0}^{|\Psi_\gamma|-1} x_{\gamma,n^\nu_i,n^s_u} \cdot \text{cpu}_{\gamma,n^\nu_i} \tag{3a}$$

$$R^{\text{pri}}_{\text{brc}}(n^s_u) = \sum_{\gamma=0}^{|\Gamma|-1} \sum_{i=0}^{|\Psi_\gamma|-1} x_{\gamma,n^\nu_i,n^s_u} \cdot \text{cpu}^{\text{brc}}_{\gamma,n^\nu_i} \tag{3b}$$

$$R^{\text{bk}}_{\text{brc}}(n^s_u) = \sum_{\gamma=0}^{|\Gamma|-1} \sum_{i=0}^{|\Psi_\gamma|-1} z_{\gamma,n^\nu_i,n^s_u} \cdot \text{cpu}^{\text{brc}}_{\gamma,n^\nu_i} \tag{3c}$$

$$R^{\text{pri}}_{\text{comp}}(n^s_u) + R^{\text{pri}}_{\text{brc}}(n^s_u) + R^{\text{bk}}_{\text{brc}}(n^s_u) + R^{\text{bk}}_{\text{comp}}(n^s_u) \leq C^{\text{cpu}}_{n^s_u} \tag{3d}$$

$$\sum_{u=0}^{|P|-1} x_{\gamma,n^\nu_i,n^s_u} = 1 \tag{4}$$

$$\sum_{u=0}^{|P|-1} z_{\gamma,n^\nu_i,n^s_u} = 1 \tag{5}$$

$R^{\text{pri}}_{\text{comp}}(n^s_u)$ and $R^{\text{pri}}_{\text{brc}}(n^s_u)$ are the computing resource demand and BRC consumed by the primary VNFs mapped on PM n^s_u, respectively. $R^{\text{bk}}_{\text{brc}}(n^s_u)$ and $R^{\text{bk}}_{\text{comp}}(n^s_u)$ are BRC and shared redundancy by the backup VNFs on PM n^s_u, respectively. Figuring out the $R^{\text{bk}}_{\text{comp}}(n^s_u)$ on each PM is the key of the problem. Eq. 4 indicates that one VNF can only be mapped on one PM, and Eq. 5 indicates that each VNF only has one backup.

The link capacity constraint is as following:

$$\sum_{\gamma=0}^{|\Gamma|-1} \sum_{(n^\nu_i,n^\nu_j)\in E^\nu_\gamma} b_{\gamma,n^\nu_i,n^\nu_j} \cdot y_{\gamma,n^\nu_i,n^\nu_j,n^s_u,n^s_v} \leq C^{\text{link}}_{(n^s_u,n^s_v)} \tag{6}$$

Finally, we should assure that the availability requirement is satisfied for each SFC γ:

$$\prod_{i=0}^{|\Psi_\gamma|-1} p_\varepsilon(n^\nu_i \leftarrow n^s_u) \geq A_\gamma \tag{7a}$$

$$p_\varepsilon(n^\nu_i \leftarrow n^s_u) = \mathcal{F}(R^{\text{bk}}_{\text{comp}}(n^s_u), \sum_{\gamma=0}^{|\Gamma|-1} \sum_{j=0}^{|\Psi_\gamma|-1} z_{\gamma,n^\nu_j,n^s_u}, \cup_{\gamma=0}^{|\Gamma|-1} \cup_{j=0}^{|\Psi_\gamma|-1} p_0(n^\nu_j) \cdot z_{\gamma,n^\nu_j,n^s_u}) \tag{7b}$$

Eq. 7b indicates that $p_\varepsilon(n^\nu_i \leftarrow n^s_u)$ is the function of shared redundancy, number of VNFs that share the same block of redundancy, and the inherent availability of all VNFs involved in the same share of redundancy.

The VNF deployment problem is NP-hard [1][2] and it is a sub-question of our problem. So our problem is NP-hard too. Besides, there are non-linear constraint (Eq. 7a) and non-analytical function relationship (Eq. 7b) in our problem. So it is almost incapable to solve the problem in theory but practical to design an efficient heuristic solution.

III. Proposed Solution

A. Framework of the solution

In our solution, we backup the VNFs based on PMs. Which is to say, firstly one PM is filled in with SFCrs as many as possible and the related VNFs are instantiated in it, then the redundancy is reserved in its nearby PM to assure the availability requirements of SFCrs in the PM. By placing the redundancy in the nearby PM, we can improve the resistance to the failure of PM empirically. So the above processes solve **Q1** and **Q2**.

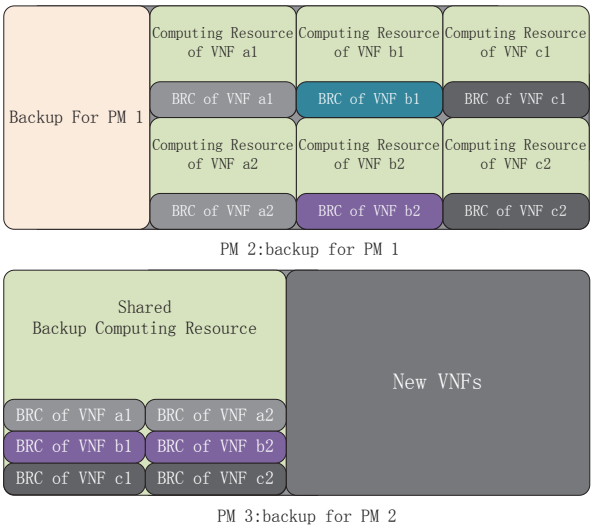


Fig. 1: A deployment and backup case

Fig. 1 shows a simple deployment and backup case. In PM 2, there are 2 deployed SFCs, which are SFC {VNF a1 $\rightarrow$ VNF b1 $\rightarrow$ VNF c1} and SFC {VNF a2 $\rightarrow$ VNF b2 $\rightarrow$ VNF c2} respectively. As the figure shows, each of the primary VNFs in PM 2 has its own BRC. To resist the failures of PM 2, the redundancy for the VNFs in PM 2 is reserved in PM 3. Then it can be found that the resource consumptions are divided into two parts for each PM, which are the redundancy for the VNFs in the adjacent PM and the computing resource consumptions for the regular functioning of deployed VNFs. And it should be figured out how much shared redundancy ($R^{\text{bk}}_{\text{comp}}(n^s_3)$) is needed to assure the availability requirements of SFCrs in PM 2. It is worthy noting that the BRC of all VNFs are reserved in the backup PM, to maintain the status of primary VNFs in a standby mode. In this way, the backup VNF can take place the functioning of failed primary VNF rapidly.

With the increasing of redundancy, the availability of each VNF in the corresponding PM increases too. Nextly, we introduce the availability modification model of VNFs based on the shared redundancy.

Algorithm 1: Modify availability

Input: Number of VNFs: N,
Shared Redundancy: φ;
Output: Modified availability of N VNFs.
for i *in* N **do**
  $p_{\varepsilon}^{i} = p_{0}^{i}$.
  $\varphi_{\delta} = \varphi - R_{c}^{i}$.
  $\vec{l_r} = (R_c^1, R_c^2, ..., R_c^{i-1}, R_c^{i+1}, ..., R_c^N)$.
  Calculate the Cartesian power of $N-1$ {0,1}, indicated as Ω.
  for $\vec{\omega}$ *in* Ω **do**
    $R_{\omega} = \vec{l_r} \cdot \vec{\omega}^T$.
    if $\varphi_{\delta} \geq R_{\omega}$ **then**
      Calculate the availability improvement, p_{δ}^{ω}.
      $p_{\varepsilon}^{i} += (1 - p_{0}^{i}) \cdot p_{\delta}^{\omega}$.
    else
      Continue.
    end
  end
end

B. Availability Modification of VNFs based on Shared Redundancy

Algorithm 1 shows the procedures of the availability modification. p_{ε}^{i} is the modified availability of VNF i, while p_{0}^{i} is the availability of VNF i without any redundancy, also can be called the inherent availability; R_{c}^{i} is the computing resource demand by VNF i; φ_{δ} is the residual redundancy after reserving resource for VNF i; R_{ω} is the resource needed to handle the failures of other VNFs except VNF i. We will demonstrate the process through a simple case in following.

Assume that there are 3 VNFs in total, namely VNF e, VNF f, and VNF g. And their computing resource demand are $R_{c}^{e} = 3$, $R_{c}^{f} = 4$, and $R_{c}^{g} = 5$ with inherent availability of $p_{0}^{e} = 0.94$, $p_{0}^{f} = 0.95$ and $p_{0}^{g} = 0.96$ respectively. Then the availability modification process of VNF e with shared computing redundancy of $\varphi = 8$ is described.

TABLE II: Availability Modification Case of VNF e

Bit(f)	Bit(g)	$\varphi_{\delta} \geq R_{\omega}^{fg}$	p_{δ}^{e}
0	0	1	0.05472
0	1	1	0.000228
1	0	1	0.000288
1	1	0	0

Firstly, two {0,1} sets for VNF f and VNF g are set up, where 0 indicates that the corresponding VNF is functioning regularly, and 1 indicates the VNF is outage. For the 2 VNFs, the Cartesian power [17] of two {0,1} sets is {(0,0),(0,1),(1,0),(1,1)}. Then the availability improvements of VNF e based on the shared redundancy need to be calculated in different occasions, which are shown in Table II. Bit(f) indicates the functioning status of VNF f, while Bit(g) is for the functioning status of VNF g. In Table II, $\varphi_{\delta} = \varphi - R_{c}^{e} = 5$. R_{ω}^{fg} is the resource needed to handle the failures of VNF f and VNF g, and it is calculated as Eq. 8. If $\varphi_{\delta} \geq R_{\omega}^{fg}$, then the failure can be handled by the residual redundancy.

$$R_{\omega}^{fg} = \mathrm{Bit}(f) \cdot R_{c}^{f} + \mathrm{Bit}(g) \cdot R_{c}^{g} \tag{8}$$

For each failure occasion, Eq. 9 is used to calculate the availability improvement of VNF e.

$$\begin{aligned} p_{\delta}^{e} =& (1 - p_{0}^{e}) \cdot [(1 - \mathrm{Bit}(e)) \cdot p_{0}^{f} + \mathrm{Bit}(f) \cdot (1 - p_{0}^{f})] \\ & \cdot [(1 - \mathrm{Bit}(g)) \cdot p_{0}^{g} + \mathrm{Bit}(g) \cdot (1 - p_{0}^{g})] \end{aligned} \tag{9}$$

After calculating the availability improvement of all occasions, the availability of VNF e given shared redundancy $\varphi = 8$ is $p_{\varepsilon}^{e} = p_{0}^{e} + 0.05472 + 0.000228 + 0.000288 = 0.995236$.

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

Based on the above availability modification model, we can get the modified availability of each VNF given a block of shared redundancy. Further, the modified availability of each SFC can be get based on Eq. 1, and the needed redundancy to assure the availability requirements of SFCs can be figured out subsequently, which solves **Q3**.

Algorithm 2: JDBSR

Sort the SFCs in Γ based on their resource demands in descending order.
i = 0.
while *1* **do**
  Start a new PM, $i += 1$.
  for γ *in* Γ **do**
    if *PM* i *can hold* γ **then**
      Map SFC γ in PM i.
      Remove SFC γ from Γ.
    else
      Continue.
    end
  end
  Check the rest SFCs and find those can be mapped in PM i.
  Backup SFCs in PM i, **Algorithm 3**.
  if Γ *is not empty* **then**
    Pick up the SFC that demands the least resource, and try to map it in PM i, **Algorithm 4**.
    Backup SFCs in PM i, **Algorithm 3**.
  else
    Break.
  end
end

C. Availability Aware SFC Deployment

Algorithm 2 along with Algorithm 3, Algorithm 4 and Algorithm 1 describes the details of our solution.

Algorithm 3: Backup PM

```
1:Input: PM i, PM i + 1
2:Output: PM i, PM i + 1
3:Redundancy, φ = 0.
4:Modify the availability of VNFs in PM i.
5:if All SFCs' availability requirements are satisfied. then
6:    return PM i, PM i + 1.
7:else
8:    while 1 do
9:        Find the VNF ω that consumes the most resource
          in PM i, and its resource consumption is
          indicated as ω_r.
10:       φ = φ + ω_r, modify the availability of all VNFs
          in PM i, Algorithm 1.
11:       Remove VNF ω from the procedure.
12:       if All SFCs' availability requirements are
          satisfied. then
13:           Break.
14:       else
15:           Continue.
16:       end
17:   end
18:end
```

In Algorithm 2, all the SFCs are traversed firstly in descending order based on their resource demand, and then they are mapped in PM i until the PM cannot hold any SFC as a whole. Then redundancy is reserved in PM $i+1$ for the SFCs in PM i (Algorithm 3). After that, we pick out the SFC that consumes the least resource, and try to map part of the SFC into the current PM, aiming to utilize the resources in the PM as much as possible (Algorithm 4).

When reserving redundancy for SFCs, the shared redundancy (φ) increases based on the computing demands of VNFs in descending order (lines 9-10 in Algorithm 3), until all SFCs' availability requirements are satisfied. In this way, we can determine the redundancy volume efficiently and quickly with a lower algorithm complexity.

Algorithm 4: Map the last sfc

```
1:Input: Last SFC γ_l,PM i,P
2:Output: PM i, P
3:for VNF in SFC γ_l do
4:    if PM i can hold the VNF then
5:        Map the VNF in PM i.
6:    else
7:        Break.
8:    end
9:end
10:Map the rest VNFs in PM i + 1 or other PMs in P as a
   whole.
```

In Algorithm 4, the VNFs in the last SFC are traversed one by one in sequence, and they are mapped into the PM until there is one VNF cannot be hosted on the PM. Even if there are other VNFs that can be mapped in the rest part of the SFC, the mapping process is still stopped, trying to keep the VNFs of each separated part in order. Because deploying one SFC across multiple PMs will incur the bandwidth consumptions between the PMs, and breaking the order of VNFs tends to result in more bandwidth consumptions.

D. Complexity

The complexity of SFC mapping process is at the level of $\mathbf{O}(|\Gamma|^2)$. However, when reserving redundancy (Algorithm 3), the computing of Cartesian power is involved (Algorithm 1), which is in exponential complexity. Assuming that there are N VNFs in one PM, then the complexity of Algorithm 1 is at the level of $\mathbf{O}(N \cdot 2^N)$. So the complexity of JDBSR also relies on the ratio between PM capacity and VNF's resource demand.

IV. Numerical Simulation

In this section, we evaluate the performance of JDBSR, compared with the benchmarks. One is the CERA in [6], in which they selected the VNF that has the largest Cost-aware Importance Measure (CIM) to backup, until the availability requirement of the SFC is satisfied. The other is CERA_oto, which is a variation of CERA, and it is a 1:1 backup scheme that reserves a backup for every VNF in one SFC.

A. Simulation Settings

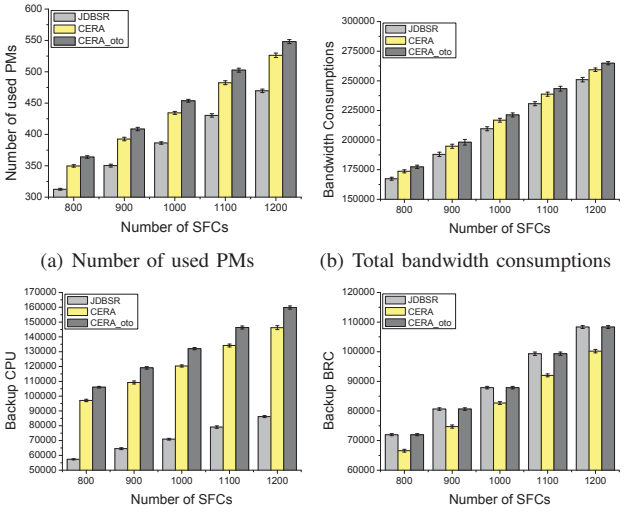
In the simulation, we have 1024 PMs in total. Each PM are associated with 1000 units CPU resource and 1000 units bandwidth resource. For the SFCs, each SFC consists of 3 to 6 VNFs, and the CPU and bandwidth demand obeys uniform distribution of (10,50). The inherent availability of the VNFs obeys uniform distribution of (0.90,0.999), and the availability requirements of the SFCs are selected randomly from [0.90,0.99,0.999,0.9999,0.99999].

B. Results

In the simulations, our solutions are compared with the benchmarks from four aspects, which are number of used PMs, total bandwidth consumptions, total backup CPU, and total backup BRC. Error bars represent the 95% confidence intervals in each figure.

Fig. 2 shows the results versus varying SFC number. From the results, we can see that JDBSR achieves better performance than CERA and CERA_oto in number of used PMs, total bandwidth consumptions, and total backup CPU, owing to resource sharing mechanism. However, for the backup BRC, CERA performs the best, and JDBSR consumes the same volume of BRC with CERA_oto. The reason is that both JDBSR and CERA_oto backup the BRC of all VNFs. JDBSR instantiates the corresponding instance for every primary VNF, however, the backup computing resources are shared among multiple primary VNFs.

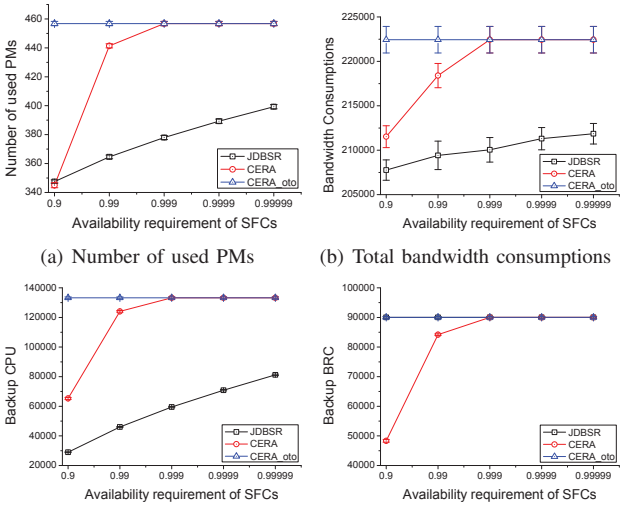
Fig. 3 shows the results versus varying availability requirements of SFCs. From the results, we can see that JDBSR still



(a) Number of used PMs (b) Total bandwidth consumptions

(c) Total backup CPU (d) Total backup BRC

Fig. 2: Performance comparisons versus varying SFC number



(a) Number of used PMs (b) Total bandwidth consumptions

(c) Total backup CPU (d) Total backup BRC

Fig. 3: Performance comparisons versus varying SFC availability requirements

has advantage over the benchmarks, especially when the availability requirement becomes stricter. For CERA, it has a good performance when the availability requirement of SFCs is lower than the inherent availability of VNFs, sometimes even a slight better than JDBSR. However, when the availability requirement of SFCs is higher than the inherent availability of VNFs, the selection process based on CIM will not work, and it will backup every VNF in one SFC to meet the availability requirement, then it degenerates into CERA_oto.

V. Conclusion

In this paper, we study the availability aware VNF deployment problem considering SFCs. The resource sharing mechanism is taken advantage of to improve the redundancy efficiency when providing backups to the primary VNFs. Then we propose a joint VNF deployment and backup scheme through shared redundancy (JDBSR). The simulation results show that JDBSR can make a better utilization of the network resources, and acquire a better performance than the benchmarks.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61671420 and No. 61672106.

References

[1] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1346–1354.

[2] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

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

[4] N. ISG, "Network functions virtualisation (nfv); reliability; report on models and features for end-to-end reliability," *ETSI GS NFV-REL*, vol. 1, p. v1, 2016.

[5] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "Grep: Guaranteeing reliability with enhanced protection in nfv," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2015, pp. 13–18.

[6] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[7] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE, 2016, pp. 128–133.

[8] S. Bijwe, F. Machida, S. Ishida, and S. Koizumi, "End-to-end reliability assurance of service chain embedding for network function virtualization," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–4.

[9] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5g mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 483–496, 2016.

[10] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and sfc requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[11] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 255–260.

[12] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 63–74.

[14] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. Chang, M. Lyu, and R. Buyya, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Transactions on Services Computing*, 2016.

[15] R. Yu, G. Xue, X. Zhang, and D. Li, "Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers," in *IEEE INFOCOM*, 2017.

[16] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma, "Application-driven bandwidth guarantees in datacenters," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 467–478.

[17] *Cartesian product*, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cartesian_product.