Guaranteeing Availability for Network Function Virtualization with Geographic Redundancy Deployment

Jingyuan Fan, Chaowen Guan, Kui Ren and Chunming Qiao
Computer Science and Engineering Department
SUNY at Buffalo, Buffalo, New York 14260
Email: {jfan5, chaoweng, kuiren, qiao}@buffalo.edu

***Abstract*—Network Function Virtualization (NFV) is a promising technique to greatly improve the effectiveness and flexibility of network services through a process named Service Function Chain (SFC) mapping, with which different network services are deployed over virtualized and shared platforms in data centers. However, such an evolution towards software-defined network functions introduces new challenges to network services which require high availability (HA), especially when catastrophic failures happen, such as earthquake and power outage, which may bring a whole data center down. One effective way of protecting the network services from such failures is to use sufficient redundancy, and in particular geographic redundancy so that when the primary VMs in a failed data center go down, the backup VMs deployed at another location can assume the role of the primary. In doing so, however, the efficiency of physical resources may be greatly decreased. To address such issue, this paper defines availability-aware SFC mapping problem and presents a novel online algorithm that can minimize the physical resources consumption while guaranteeing the required HA within polynomial time. Theoretically, we prove the intractability of the problem and our backup VNFs picking approximation algorithm can achieve a normalized relative error of $\left((e-1) \times AE(OPT) + AE(\emptyset)\right)/e$, where $AE(OPT)$ and $AE(\emptyset)$ are the availabilities with an optimal backup solution and without backups, respectively. Simulation results show that our proposed algorithm can significantly improve SFC request acceptance ratio and reduce resource consumption.**

I. Introduction

Network Function Virtualization (NFV) is a driving force behind implementing network functions such as deep packet inspection (DPI) and PDN gateway (PGW) in software-based processing functions that run on the standardized commodity storage, servers and switches. NFV enables a virtualized and shared platform that can significantly reduce the hardware cost and investment, as well as greatly improve the efficiency and flexibility of utilizing physical hardware resources. Since NFV can bring benefits in terms of manageability and flexibility for service providers, many of them are starting to put this technology into practice. One motivating case is the AT&T Integrated Cloud (AIC), which are basically data centers where virtualized functions can be run on common off-the-shelf hardware platform. According to [1], 29 AIC nodes have been deployed across the US, and at least 40 more nodes will be added by the end of this year. By the end of 2020, they plan to virtualize 75 percent of their massive network. Network functions are deployed on the shared platform through a process called *Service Function Chain* (SFC) mapping defined by current standardization efforts [3]–[5]. A SFC consists of a set of *Virtual Network Functions* (VNFs) interconnected by logical links. Multiple SFCs from distinct clients may share the computing and networking resources in order to improve the resource utilization. For service providers, it is essential to find an optimal mechanism for such SFC mapping so that physical resources can be efficiently used.

However, software-based network functions introduce unique availability challenges to future networks. Managing availability usually implies a service recovery mechanism after faults are detected. Traditional ways of improving availability by utilizing diversity may no longer work since we use uniform platform architecture in NFV.

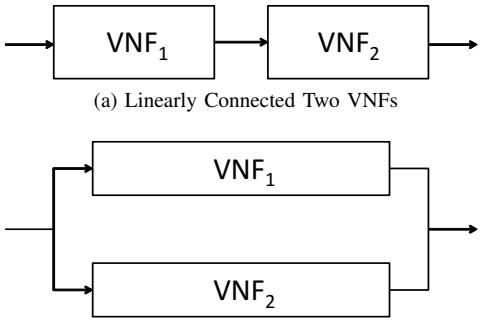
Ensuring availability in multisite scenario, which refers to the cases when VNFs are deployed on multiple Virtual Infrastructure Managers (VIMs), is especially important and challenging. In particular, virtual machines (VMs) used for a service chain may be distributed at geographically different locations for some practical reasons [2], [3]. However, when catastrophic failures such as flood, earthquake and propagating software faults happen at one site, there can be serious consequences. Recently, several services of Amazon and Google suffered outages due to software problems spanned over the whole data center and lightening in the US and Europe respectively [6], [7]. In NFV, as network functions are virtualized and run on VMs, when an application fails, restarting it repairs the application and thus can continue providing the service again. Restarting may take a significant amount of time causing service outage, for which redundancy is a *de-facto* technique adopted by industry in masking such failures [12], [14], [15]. When the service is protected by redundancy, the service can be failed over to the standby entity deployed at another location, which replaces the failed one while it is being repaired. Specifically, VNFs are deployed in multiple sites, which are geographically separated and are managed by separate VIMs. When such a catastrophic failure happens, the VNFs at the failed site can quickly fail over to the redundant one so as to proceed the service. It is recommended, according to ETSI [8], off-site resources should be available for enterprise or large scale customers and network operators

service traffic for data services recovery, and most redundant resources should be off-site for general consumer public and Internet service provider (ISP) traffic. Since a service is considered available only when all the functions it requires are available, how to map functions onto physical substrate in a resource-efficient way while ensuring availability is critical and challenging in NFV.

In this paper, we take the first step by addressing the following problem of availability-aware SFC mapping: *what is the minimum number of backup VNFs service provider needs to provision to guarantee a certain degree of availability? What is the best protection strategy in terms of availability improvement and resource consumption? Furthermore, how to map both primary and backup VNFs and interconnecting logical links in a resource-efficient way?* The goal is to use the least amount of resources to meet each request's availability requirement such that a higher SFC request acceptance ratio can be achieved, while reducing the resource consumption for service providers. Hence, developing an effective protection mechanism and an efficient SFC mapping scheme to meet different clients' *Service Level Agreement* (SLA) (e.g., the availability requirement) are essential while consuming a small amount of physical resources. To solve the availability-aware SFC mapping problem, we propose an online algorithm to guarantee each client's availability requirement while minimizing the amount of resources allocated.

Note that the availability-aware SFC mapping problem is more difficult than the problem of survivable virtual infrastructure mapping studied earlier in [9]–[11] for two reasons: 1) we need to consider network function restrictions; 2) we need an effective algorithm to search for and evaluate an efficient backup plan to meet specific availability requirements of heterogeneous SFC requests. There also exist some works [24]–[26] focusing on how to allocate the VMs among geo-distributed data centers in order to offer tolerance against physical component failures. However, in our paper, we assume that different VNFs are heterogeneous in terms of both their functional and resource requirements hence we need to consider both the functional and resource capabilities of the substrate system. Finally, to the best of our knowledge, none of these existing works has considered problems similar to availability-aware SFC mapping problem with different clients having different availability requirement and different data centers having different availability. In summary, we list the main contributions as follows:

- We propose a novel enhanced *Joint Protection* (JP) approach, and demonstrate its advantages by comparing with traditional *Dedicated Protection* (DP) and *Shared Protection* (SP) in terms of acceptance ratio performance and resource consumption.
- We for the first time prove that there's no polynomial time algorithm for solving the proposed availability-aware SFC mapping problem.
- We develop an estimation method for computing SFC availability with a polynomial time complexity, and show the estimation error is negligible.



(a) Linearly Connected Two VNFs

(b) Parallelly Connected Two VNFs

Fig. 1. Two ways of combining VNFs

- We propose a novel backup selection strategy, prove its approximation ratio, and illustrate that it can save the number of backup VNF by 42%.

The rest of the paper is organized as follows. Section II describes our availability model and protection schemes. Section III defines the problem of availability-aware SFC mapping and analyzes its complexity. Section IV introduces a polynomial running time algorithm for availability evaluation and an approximation algorithm with a theoretical lower bound for backup VNF selection. We evaluate the performance of the proposed algorithm in Section V followed by conclusion in Section VI.

II. Availability Model and Protection Schemes

A. Availability Model

To use a availability model for a given NFV deployment, we first need to model the logical structure of the system. Note that since in this paper, we are mostly concerned with ensuring service availability with geographic redundancy in multisite scenario, we assume that there are some protection schemes used in a data center so that VNFs deployed in a data center have 100% availability when the data center is working fine, so the only source resulting in service disruptions is data center being down due to catastrophic failures.

1) Availability of one single component: The availability of a complex system such as an NFV deployment can be modelled by decomposing it into constituent components [8], of which the availability are known. The availability of a component is the relative share of time the component is functioning, and thus the probability to find the component working if checking it at a random point in time. For a data center, which is a repairable component, its availability can be expressed using uptime followed by downtime, which can be characterized in terms of *Mean Time Between Failures* (MTBF) and *Mean Time To Repair* (MTTR), respectively. In gernal, the availability of a data center can be characterized as

$$A = \frac{Uptime}{Uptime + Downtime} = \frac{MTBF}{MTBF + MTTR} \quad (1)$$

2) Availability of composed system: A SFC is generally composed of a number of VNFs. In order to estimate the availability of such composite system, which is derived from the individual components it consists of, two basic ways of combining components, serial and parallel, need to be understood. In SFC, individual components are connected in a serial manner, which means that in order for the SFC to function, all components that the SFC comprises need to function at the same time. For example, as shown in Fig. 1 (a), in order for the function provided by this SFC to be available, both VNF_1 and VNF_2 need to be available at a given time. Therefore, the availability of this SFC request is:

$$A_{SFC} = A_{VNF_1} \times A_{VNF_2} \tag{2}$$

where A_{VNF_1} and A_{VNF_2} are independent because they are deployed in different data centers.

While, if two individual components are connected in a parallel way, as shown in Fig. 1 (b), it assumes that these two components are fully redundant. If both VNF_1 and VNF_2 can provide the same function, then the requested service is available when at least one of these two independent components can function, assuming there is no service disruption due to fail over. Thus, the availability of this SFC request can be described as:

$$A_{SFC} = 1 - ((1 - A_{VNF_1}) \times (1 - A_{VNF_2})) \tag{3}$$

Using these two basic models, we can model and evaluate the availability of a more complicated SFC request with different protections schemes, which will be discussed in the next subsection. Note that this model can be applied to both VNF failures and link failures.

B. Protection Schemes

In this subsection, we will first describe two traditional protection schemes that are widely used in industry and studied in academia, then we will propose a new one called *Joint Protection* (JP).

1) Dedicated Protection: DP is one of the most popular protection methods adopted in many system [12], [14]. Usually for DP there is one active VNF and one backup VNF so that when the data center holding the active VNF fails, the traffic can be diverted to the backup VNF to continue providing service. As shown in Fig. 2 (a), backup b_i and b_j will duplicate the functionality of n_i and n_j respectively, and connect to their neighboring VNFs (e.g., n_{i-1} and n_{i+1}, n_{j-1} and n_{j+1}). Although it provides high availability, it results in high resource usage.

2) Shared Protection: To reduce resource consumption, researchers have proposed another protection scheme called shared protection. In SP, there are multiple (usually two) active VNFs and one backup VNF that can take over when any one of the data centers holding the active VNFs fails. It saves resources by allocating max(s_{n_i}, s_{n_j}) resources on backup b to protect either n_i or n_j and connecting the backup to the neighboring VNFs of n_i and n_j shown in Fig. 2 (b), but the network can fail when both of the VNFs protected by one backup fail. Compared with DP, SP cannot provide the same level of protection.

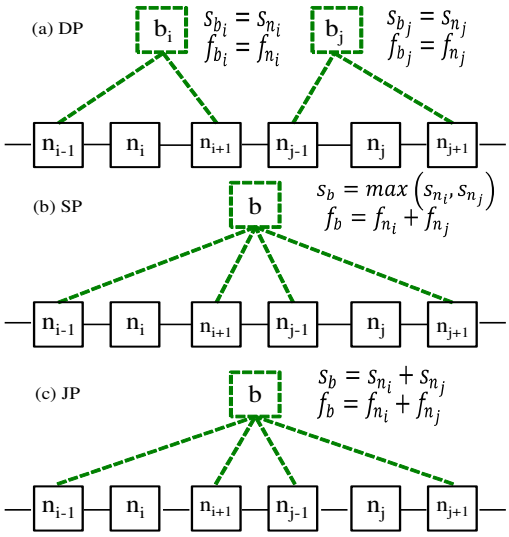


Fig. 2. Different Protection Schemes

3) Joint Protection: In this paper, we propose a novel protection strategy called *Joint Protection* (JP) for providing geographic redundancy so that even when multiple active VNFs fail at the same time, the backup VNF can provide all the functions that all the failed active VNFs provide. For example, as shown in Fig. 2(c), JP provides protection to both VNFs connected to it, and the amount of backup resources reserved at VNF b will be sufficient for both n_i and n_j (i.e., $s_b = s_{n_i} + s_{n_j}$). Thus, the SFC can still function normally even if both n_i and n_j fail simultaneously. As a result, one in JP can provide high availability as in DP while potentially saving resources as in SP. Our proposition will be validated via simulations in Section 5.

III. Problem formulation & Complexity Analysis

In this section, we formally describe the availability-aware SFC mapping problem, the failure models we consider in the paper and show the hardness of the problem.

A. availability-aware SFC Mapping Problem

We consider our problem with a generic network model. Given a *Physical Substrate* (PS) $P_s(N_s, L_s)$, where N_s is the set of *Physical Nodes* (PNs) and L_s is the set of *Physical Links* (PLs). For each PN $n \in N_s$, it is associated with a set of k types of resources $S_n^k = \{s_n^i | i \in [1, k]\}$, where s_n^i denotes the capacity of resource of type i. In addition, each PN n is associated with availability A_n. Given the set of resources available at a PN n, it can provide a set of functions denoted by f_n^P. $F_n = \bigcup_{i=1}^{N_s} f_i^P$ is the set of all functions that the network can provide. For example, PN F, as shown in Fig. 1, can only provide functions for *Home Subscriber Server* (HSS), and it's availability is 0.94, while PN E can provide functions for PDN/Serving Gateway and

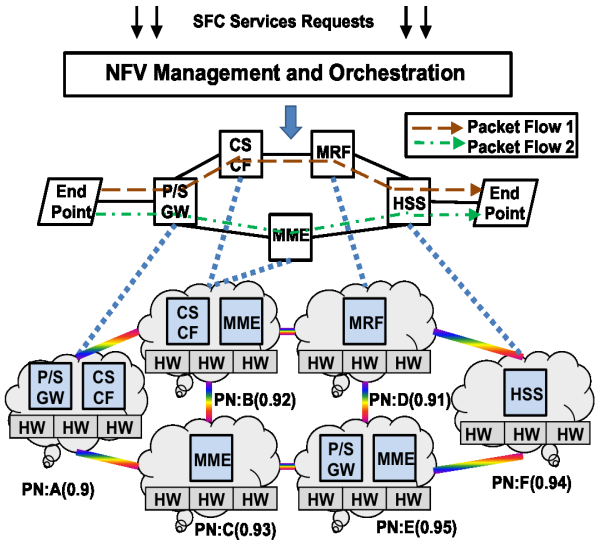


Fig. 3. NFV Architecture

Mobility Management Entity (MME). For a given SFC request r denoted by $V_r(N_r, L_r, \theta_r)$, $N_r = \{n_r^1, n_r^2 \dots n_r^{|N_r|}\}$ is the set of VNFs, each of which requires a set of resources to perform one single network function $f_{n_r^j} \in F_n$. $F_r = \bigcup_j f_{n_r^j}$ is the set of functions that request r needs. Each logical links $l_r \in L_r$ has a bandwidth demand and θ_r is the availability requirement of this SFC. To map a SFC, we not only need to map a VNF to a PN by reserving an appropriate type and amount of resources in the chosen PN to perform the function requested by that VNF, but also map a logical link through allocating an appropriate amount of bandwidth along each and every physical link over the chosen path to carry the traffic flow from one VNF to another VNF. For instance, in Fig. 3, the VNF requiring *Media Resource Function* (MRF) on packet flow 1 can only be mapped onto PN D as it's the only PN providing such function, while the traffic from MRF to HSS can flow from PN D to F directly or redirect to any other path starting with PN D and ending with PN F.

As discussed in the previous section, a service is considered being available at a given time if all the functions the service requests are able to function normally. In this work, since we mainly focus on service protection when catastrophic failures bring data centers down, we only consider node failures. When no protection is provisioned, the availability of a SFC request r can be obtained as $A_r = \prod_{f \in F_r} A_f$, where A_f is the availability of the PN providing function f. However, when there are protections, evaluating availability becomes a hard problem (discussed in Section III.B). Upon mapping, a SFC request is considered as being blocked if any VNFs or logical links cannot be mapped or the availability cannot meet the client's requirement. Therefore, we can define the SFC availability-aware mapping problem as follows. Given a set of SFC requests, each with a specific availability requirement, we need to find out the minimum number of backup VNFs needed in order to achieve each availability requirement and efficiently

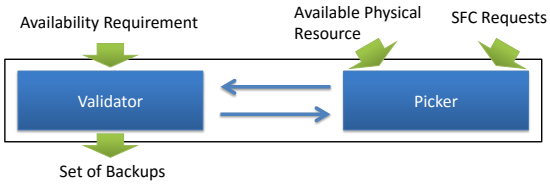


Fig. 4. Proposed Architecture

assign primary and backup VNFs to physical nodes. Noted that we try to solve a more general problem without assuming the number of failed data centers at the same time. A decision algorithm can be embedded in a centralized system, such as NFV Management & Orchestration (MANO), that manages all the incoming requests.

B. Complexity Analysis

In this section, we will briefly discuss the complexity of our problem. Due to the limit of pages, we defer more details of the complexity analysis to **Appendix A**. Concretely, below we list the conclusions we draw from the analysis.

Theorem 1. *The problem of verifying if the availability is above a given threshold (denoted by VA) is PP-complete.*

Even with an oracle to the VA problem, we still cannot optimally decide if there exists a solution for a SFC request,and finding a local optimal is difficult.

Theorem 2. *Determining if there exists a solution for a SFC request (denoted by DE) is NP^{PP}-complete.*

Theorem 3. *Finding a local optimal solution for a SFC request (denoted by LM) is co-NP^{PP}-complete.*

The objective of globally minimizing the number of backup VNFs further elevates the complexity.

Theorem 4. *Finding a maximum set A belongs to $NP^{NP^{PP}}$.*

The complexity classes we mention satisfy these containment properties and relations to other classes [16]:

$$\mathrm{P} \subseteq \begin{matrix}\mathrm{NP}\\ \text{co-NP}\end{matrix} \subseteq \mathrm{PP} \subseteq \begin{matrix}\mathrm{NP^{PP}}\\ \text{co-NP}^{\mathrm{PP}}\end{matrix} \subseteq \mathrm{NP}^{\mathrm{NP}^{\mathrm{PP}}} \subseteq \mathrm{PSPACE}$$

Hence the availability-aware SFC mapping problem is believed to be intractable.

IV. Algorithm Design

In this section, we propose an online algorithm for providing geographic redundancy for NFV. The basic components are shown in Figure 4. Our design is independent of the specific topology used by the physical network or SFC. Our goad is to find the most resource-efficient mapping for each SFC request while meeting its availability requirement. Inputs to the decision algorithm are the physical topology, including resource availability for each PN and PL, and the mapping of each SFC request without any backups and the availability

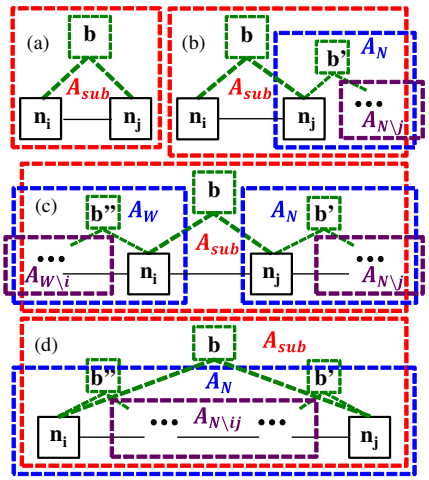


Fig. 5. SFC Availability Evaluation

requirement of each SFC request. The mapping process of SFC requests is same as [13]. The metric of interest is the SFC acceptance ratio, defined by the number of accepted SFC request by the decision algorithm over the total number of SFC request. The protection scheme used here is the JP we proposed in Section II.B.

Based on our complexity analysis in Section III.B, we know finding the optimal solution is challenging. Hence, to address the challenges, we can decompose the problem into two sub-parts: A 'backup picker' (Section IV.B) proposes backup VNFs selection that will maximize the availability, subject to the resource availability for each PN and PL. A 'backup validator' (Section IV.A) confirms or reject the proposal by evaluating the availability and comparing the evaluation with the SFC requirement. In addition, the validator also needs to update availability for each PN after each time picker chooses backups so that the picker can better propose backup for subsequent iterations. In each iteration, the picker chooses one backup VNF for two primary VNFs. The process repeats until either the availability requirement is met, confirmed by the validator, or no more physical resource is available to improve the SFC's availability, output by the picker.

A. Validating Backup Choice

The validator determines whether the backups proposed by the picker are enough to meet the availability requirement. As seen from **Theorem. 1**, the problem VA is PP-complete, so there's no polynomial time solution to solve the problem. Previous research [17], [18] proposed solutions using *Monte Carlo* related methods, but it's hard to determine how many steps are needed to converge to the stationary distribution within an acceptable error, and the procedure is time consuming. In this subsection, we will show a computational easier way for problem VA. Let us explain the rationale first. We consider the whole network as a composition of several independent sub-networks, and thus, the availability of the request is the multiplication of the availability of each sub-network as discussed in Section II.A. At beginning, all VNFs are considered as separate sub-networks, so the availability of the SFC request is the multiplication of the availability of the physical nodes that primary VNFs are mapped to. Until the availability requirement is met, two primary VNFs with a backup VNF are selected for each iteration by the picker. We explain in Section IV.B which two VNFs are selected. Then, a new sub-network is formed, composed of the backup VNF and two sub-networks, each of which contains one of the two selected primary VNFs. The availability of these two selected VNFs and this new sub-network need to be updated. Then the availability of this new sub-network will be used in evaluating the the availability of the SFC request.

We next show how to evaluate the availability of the new sub-network using an example. Assume the availability of the two selected VNFs n_i, n_j are A_i, A_j, and the availability of the backup b is A_b. n_i and n_j provides function f_{n_i}, f_{n_j} respectively. Before provisioning b as backup there are totally four possible cases:

1) Neither n_i nor n_j has backups: If neither of the two selected VNFs has backups, the availability of the sub-network they form equals to the probability that the backup is availability or both of the two selected VNFs are availability while the backup cannot work.

$$A_{sub} = 1 - (1 - A_b) \times (1 - A_i A_j) \tag{4}$$

2) Only one of n_i or n_j has backups: Without loss of generality, we assume n_j is the one already has backups, and we denote the sub-network contains n_j as N. We analyze this case by considering which sub-network provides the function that n_j requires. We can observe that this function is provisioned either by the sub-network N or the new backup b, and these two situations are mutually excluded. So the new sub-network is considered working if the sub-network N functions properly and at least one of the backup b and n_i works properly, or backup b is available and all the nodes in sub-network N except for the one providing f_{n_j} are available. Therefore,

$$A_{sub} = A_N \times (1 - (1 - A_b)(1 - A_i)) + A_b \times A_{N\setminus j} \tag{5}$$

where $A_{N\setminus j}$ is the probability that sub-network N can provide all the functions except f_{n_j}. To compute $A_{N\setminus j}$, we decompose $A_N = r'_N - A_{N\setminus j} = A'_N - (1 - A_j)\prod_{k=1}^{M}(1 - A_{j_k})A''_N$, where A'_N is the probability that the sub-network N may or may not provide f_{n_j}, while all the others function normally, A''_N is the probability that all functions except f_{n_j} in sub-network N are available, A_{j_k} is the kth backup connected with n_j excluding backup b, and M is the total number of backups that n_j currently has. We then define $\tau = \frac{A'_N}{A''_N} \approx 1 + \epsilon$, where ϵ is a small constant. Noted that as the sub-network contains more nodes, τ gets closer to 1. So

$$A_{N\setminus j} = \frac{(1 - A_j)\prod_{k=1}^{M}(1 - A_{j_k})A_N}{\tau - (1 - A_j)\prod_{k=1}^{M}(1 - A_{j_k})} \tag{6}$$

3) Both n_i and n_j have backups and they belong to different sub-networks: Since n_i and n_j belong to different sub-networks, we denote the sub-networks containing n_i and n_j using W and N respectively. Applying similar strategy described in the previous case, the probability that the new sub-network functions can be decomposed into two cases: when either both sub-networks W and N work properly, or when backup b is available and at least one of the sub-networks W and N fails to provide f_{n_i} and f_{n_j} while other functions can be provisioned. Hence,

$$A_{sub} = A_N A_W + A_b(A_{N\setminus j}A_{W\setminus i} + A_{N\setminus j}A_W + A_{W\setminus i}A_N) \tag{7}$$

4) Both n_i and n_j have backups and they belong to the same sub-network: Similarly, when the sub-network N containing n_i and n_j both works properly, whether or not backup b can work has no influence on the availability of the sub-network. And when backup b is available, at most one of the functions f_{n_i} and f_{n_j} should be available. Thus,

$$A_{sub} = A_N + A_b \times A_{N\setminus ij} \tag{8}$$

where $A_{N\setminus ij}$ denotes the probability that sub-network N can provide all functions except for f_{n_i} and f_{n_j}. Here, verifying if n_i and n_j have common backup VNFs is necessary to avoid double calculating.

Fig. 5 illustrates all the four cases for evaluating SFC availability. After updating the availability of the new sub-network, the availability of the two selected VNFs needs to be updated accordingly as well. As seen from the methods described in this subsection, for each iteration the computation complexity of computing availability is polynomial time with respect to the number of backup a VNF has. We will later show in the simulation that the estimation error is small enough to be neglected.

B. Choosing Backup

The next question to answer is how picker selects backups to minimize the number of backups that a request require. We propose a heuristic here to select backup nodes to maximize the availability that a SFC request can achieve, denoted as problem MA. The *Proposition* immediately follows [16].

Proposition 1. *MA cannot be approximated within any fixed factor in polynomial time unless P=NP.*

The intractability result holds in general, i.e., when no further constraints are put on the problem instances. Therefore, solving this problem is also hard. Here we propose a greedy heuristic algorithm based on the following *Theorem*. First we define *improvement ratio*.

Definition 1. *Define an* **improvement ratio** *as the ratio of the improvement of the network overall availability to the network overall availability before adding a backup.*

Theorem 5. *Provisioning a backup VNF to two primary VNFs whose availabilities are among the lowest maximizes the improvement ratio for each case described in Section IV.A.*

We have four cases as described in the previous section, here we only prove **Theorem. 5** for the second case. Note that we can similarly prove **Theorem. 5** for other cases as well (in fact, case 1 is simpler, and cases 3 and 4 are based on case 2).

Proof. Given that two VNFs n_i and n_j are selected, b as a backup VNF, and n_j already has backups. As n_j already has backup, we must have computed the availability of the sub-network N which contains n_j already. Then the availability of the whole network before connecting n_i and n_j with backup b is $A_{before} = A_1A_2\dots A_kA_iA_NA_p\dots A_q$, and the availability after adding backup is $A_{after} = A_1A_2\dots A_k(A_N(1-(1-A_b)(1-A_i)) + A_bA_{N\setminus j})A_p\dots A_q$ where $A_1A_2\dots A_k$ and $A_p\dots A_q$ are the availability of the sub-networks not selected, and A_N is dependent on A_j. The improvement ratio u is,

$$u = \frac{A_{after} - A_{before}}{A_{before}} = -A_b + \frac{A_b}{A_iA_N}(A_N + A_{N\setminus j}) \tag{9}$$

Let's substitute Eq. (6) for $A_{N\setminus j}$, and let $S = (1-A_j)\prod_{k=1}^{M}(1-A_{j_k})$ then calculate the partial derivative with respect to A_i and A_j respectively,

$$u_{A_i} = \frac{\partial u}{\partial A_i} = -\frac{A_b}{A_i^2A_N}(A_N + A_{N\setminus j}) \tag{10}$$

$$u_{A_j} = \frac{A_b}{A_i}\frac{(S'A_N^2 + 2\frac{\partial A_N}{\partial A_j}S\times A_N)(\tau - S) - (\tau - S)'S\times A_N^2}{(\tau - S)^2} \tag{11}$$

As the availability is always greater than or equal to 0, and $A_i \in [0,1]$, from Eq. (10) we can easily tell that u decreases monotonically as A_i increases. While the monotonic property of Eq. (11) is not as obvious as Eq. (10). To see that, we let $u_{A_j} = 0$ to compute the critical point, and get

$$2\frac{\partial A_N}{\partial A_j} = (\frac{S}{\tau - S} + 1)\frac{A_N}{1 - A_j} \tag{12}$$

Solve this partial differential equation, and get

$$A_N = \sqrt{\frac{\frac{\tau}{\prod_{k=1}^{M}(1-A_{j_k})} - (1-A_j)}{1-A_j}} \tag{13}$$

which means that when the equation holds true, we get the critical point. However, $\frac{\tau}{\prod_{k=1}^{M}(1-A_{j_k})} \gg 1$ since $\tau > 1$, $M \geq 1$ and both $A_N \in [0,1]$ and $A_j \in [0,1]$. Therefore this equation can never hold, which means u is a monotone function respect to A_j in its domain. Also we can easily check $u_{A_j=1} < u_{A_j=0}$, so we can come to the same conclusion for A_i that u decreases monotonically as A_j increases. Therefore, selecting two VNFs with lowest availabilities leads to the largest improvement ratio. □

Define $AE(B)$ as the function to calculate the availability of a request with a set of backup nodes B, and $\rho_b(B) = AE(B\cup\{b\}) - AE(B)$ as the availability improvement when adding a backup node b. So $AE(\emptyset)$ is the availability of the request without any backups. Here we

assume function AE can accurately evaluate the availability. As there are four cases in total for evaluating availability for each step, we will first prove that when a backup is provisioned for two nodes that neither of them has backup (case 1), we can have the largest availability improvement. Define ρ_b^i as the availability improvement when a backup b is provisioned and the relationship of the two nodes that backup b protects belongs to case i. i is the case number as defined in Section IV.B. Then we have the following *Lemma*.

Lemma 1. $\rho_b^1 > \rho_b^2 > \rho_b^3 > \rho_b^4$

Proof. Here we only prove the first inequality. One can prove the rest using the same method. Given a network which consists of three sub-networks, N_1, N_2 and N_3. In N_1 and N_2 all nodes are primary VNFs while N_3 is composed of primary and backup VNFs. Either we can provide a backup b for node i and j or node i and p, where node i, j and p are primary VNFs in sub-network N_1, N_2 and N_3, respectively. If node i and j are selected, then $\rho_b^1 = (A_b - A_b A_{N_1} A_{N_2}) \times A_{N_3}$; if node i and p are selected, then $\rho_b^2 = (A_{N_3} - A_{N_3} A_{N_2} + A_{N_3 \setminus p}) \times A_b A_{N_1}$. For $\rho_b^1 > \rho_b^2$, the following inequality must hold

$$\frac{1 - A_{N_1}}{A_{N_1}} > \frac{A_{N_3 \setminus p}}{A_{N_3}} \tag{14}$$

We argue that this inequality should hold in practice where a whole data center being down happens rarely, which makes $1 - A_{N_1}$ is at least one order of magnitude larger than $A_{N_3 \setminus p}$, while A_{N_1} and A_{N_3} are about the same order. □

Together with **Theorem 5**, we outline how picker select backup VNFs with the following *Theorem*.

Theorem 6. *Selecting two VNFs whose relationship belongs to the category in Section IV.B with the smallest case number, and settling ties by choosing the nodes whose availabilities are among the lowest maximizes the availability improvement for each iteration.*

Proof. Based on *Theorem 5* and *Lemma 1*, the theorem follows immediately. □

Next, we analyze how close the availability derived from the backup plan selected by picker is to the one achieved by the optimal backup solution. Assume the request has n primary nodes, and K is the number of backup nodes that can be provisioned.

Theorem 7. *When $K \leq \lceil \frac{n}{2} \rceil$, our greedy backup selection method can achieve the optimal solution.*

Proof. When $K \leq \lceil \frac{n}{2} \rceil$, or $K = \lceil \frac{n}{2} \rceil$ and n is even, according to **Lemma 1**, the nodes that each one of K backup protects should be mutually exclusive. To prove the optimality of this greedy algorithm, we need to prove the greedy choice property and the optimal structure property [23].

Greedy Choice Property: The greedy algorithm selects two nodes with the lowest availabilities among all primary nodes as the first pair of nodes to be provisioned with a backup. Say these two nodes are i and j, and the backup node is b. We have to show that there exists an optimal backup strategy that also contains a backup node to protect this pair of nodes. There are four possible cases:

1) The optimal backup strategy contains a backup node to protect node i and j, then we are done.
2) The optimal backup strategy doesn't contain any backup node to protect either of node i and j. Then we can remove any backup node from the optimal strategy and add a backup to protect node i and j. In doing so, we get a higher availability. Contradiction.
3) The optimal backup strategy contains a backup node to protect one of node i and j. Similarly, we can remove this backup node from the optimal strategy and add a backup to protect node i and j so that we can get a higher availability. Contradiction.
4) The optimal backup strategy contains two backup nodes to protect node i and j respectively. Assume node i and p is one pair and node j and q is the other pair. Without loss of generality, we assume $A_i < A_j < A_p A_q$, then we need to show when i and j form a pair and p and q form another pair, we have a higher overall availability. Based on case 1 in Section IV.B, if we can get a higher availability, then

$$\begin{aligned}
&(A_b + (1 - A_b) A_i A_j)(A_b + (1 - A_b) A_p A_q) > \\
&\quad (A_b + (1 - A_b) A_i A_p)(A_b + (1 - A_b) A_j A_q) \\
&\quad \Leftrightarrow A_i A_j + A_p A_q > A_i A_p + A_j A_q \\
&\quad \Leftrightarrow A_p (A_q - A_i) > A_j (A_q - A_i)
\end{aligned} \tag{15}$$

Since $A_q > A_i$ and $A_p > A_j$ by our assumption, the inequality holds. Contradiction.

Optimal Structure property: Let P_1 be the subproblem obtained from the original problem P by removing node i and j. Let S be an optimal backup strategy for the original problem P, in which node i and j are the first pair of node that gets protected. Let S_1 be obtained from S by deleting b. Then S_1 is a backup strategy for the subproblem P_1. We need to show that S_1 is an optimal solution for P_1. Towards a contradiction, suppose this is not the case. We replace the backup strategy in S except b by the optimal backup strategy of P_1, we get another backup strategy S' of P with a higher availability. This contradicts the fact that S is an optimal strategy of P.

Since both properties hold, the greedy algorithm is correct.

When $K = \lceil \frac{n}{2} \rceil$ and n is odd, after we choose $\lfloor \frac{n}{2} \rfloor$ pairs of nodes to provide with a backup each, we achieve the maximum availability possible. Then we greedily find the node with a backup protected and the lowest availability and pair it with the only left primary node which doesn't have a backup to provide them a with backup, which can maximize the availability based on **Theorem 5**. This concludes the proof. □

With $K > \lceil \frac{n}{2} \rceil$, we can prove our algorithm is near-optimal.

Theorem 8. *when $K > \lceil \frac{n}{2} \rceil$, the algorithm computes a backup scheme which maximize the availability with a normalized relative error of $\frac{e-1}{e} AE(OPT) + \frac{1}{e} AE(\emptyset)$, where $AE(OPT)$*

and $AE(\emptyset)$ are the availabilities with an optimal backup solution and without backups, respectively.

Proof. For arbitrary backup set S and T with $T - S = \{j_1, j_2, \ldots, j_\tau\}$ and $S - T = \{k_1, k_2, \ldots, k_\nu\}$, we have

$$AE(S \cup T) - AE(S) = \sum_{t=1}^{\tau}[AE(S \cup \{j_1, j_2, \ldots, j_t\}) - AE(S \cup \{j_1, j_2, \ldots, j_{t-1}\})] = \sum_{t=1}^{\tau} \rho_{j_t}(S \cup \{j_1, j_2, \ldots, j_{t-1}\}) \leq \sum_{v \in T-S} \rho_v^{\varphi(S')}(S') \quad (16)$$

where $\varphi(S')$ is the smallest case number of calculating ρ_{j_1}, and and this can be realized by dynamically changing S'. When adding an element, say j_p, into the backup set, if the way of calculating the availability is the same as the way of calculating availability when adding the 1^{st} element, then $S' = S \cup \{j_1, j_2, \ldots, j_{\nu-1}\}$; otherwise, $S' = S \cup \{j_1, j_2, \ldots, j_{\nu-1}\} - \{Q\}$, where $Q \subseteq S \cup \{j_1, j_2, \ldots, j_{\nu-1}\}$ so as to ensure the availability is calculated using the same case number when adding the 1^{st} backup node. Based on **Lemma 1**, this inequality holds.

Similarly,

$$AE(S \cup T) - AE(T) = \sum_{t=1}^{\nu}[AE(T \cup \{k_1, k_2, \ldots, j_t\}) - AE(T \cup \{k_1, k_2, \ldots, k_{t-1}\})] = \sum_{t=1}^{\nu} \rho_{k_t}(T \cup \{k_1, k_2, \ldots, k_{t-1}\} - k_t) \geq \sum_{v \in S-T} \rho_v^{\varphi(\chi)}(\chi) \quad (17)$$

where $\chi = T \cup S - \{v\}$ and $v \in S - T$. Subtracting Eq. (17) from Eq. (16), we get

$$AE(T) - AE(S) \leq \sum_{v \in T-S} \rho_j^{\varphi(S')}(S') - \sum_{v \in S-T} \rho_j^{\varphi(\chi)}(\chi) \leq \sum_{v \in T-S} \rho_j^{\varphi(S')}(S') \quad (18)$$

since the improvement of availability is always greater than or equal to 0. Taking T as the optimal solution, S to be the set S^t generated after t iterations of the greedy algorithm, and using

$$AE(S^t) = AE(\emptyset) + \sum_{i=0}^{t-1} \rho_i \quad (19)$$

and $AE(OPT) = AE(T)$, $|T - S^t| \leq K$, we have

$$AE(OPT) \leq AE(\emptyset) + \sum_{i=0}^{t-1} \rho_i + \sum_{v \in T-S^t} \rho_v^{\varphi(S'')}(S'') \quad (20)$$

where S'' is constructed from S^t in the same way as constructing S' from S. Now take $t = 0$, we have

$$AE(OPT) \leq AE(\emptyset) + K \times \rho_{max} \quad (21)$$

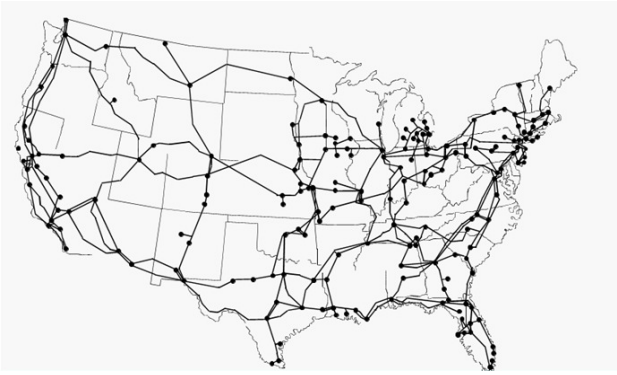


Fig. 6. Network map from a major tier-1 service provider

where $\rho_{max} = \max(\rho_v^{\varphi(S'')}(S''))$, $\forall v \in T - S^t$. So,

$$AE(OPT) - AE(\emptyset) \leq K\rho_{max} \leq K(AE(\Phi) - AE(\emptyset)) \quad (22)$$

where $AE(\Phi)$ is the availability of a solution constructed by our greedy algorithm. $\rho_{max} \leq AE(\Phi) - AE(\emptyset)$ holds based on **Lemma 1** and **Theorem 8**. So we have,

$$\frac{AE(OPT) - AE(\Phi)}{AE(OPT) - AE(\emptyset)} \leq \frac{K-1}{K} \quad (23)$$

Since $(\frac{K-1}{K})^K \leq e^{-1}$ [22], we have the approximation ratio

$$\frac{e-1}{e} AE(OPT) + \frac{1}{e} AE(\emptyset) \leq AE(\Phi) \quad (24)$$

□

V. Evaluation

In this section, we use synthetic policy to evaluate our algorithm in terms of (i) SFC request acceptance ratio, (ii) backup resource consumed by requests, and (iii) accuracy of availability evaluation proposed in Section IV.A.

A. Experimental Workloads

1) Physical Network: For physical networks, we use the network map from a major tier-1 service provider [19], as shown in Fig. 6. It has 116 nodes and 151 fiber links. Each node of the network represents one single data center, which can provide three types of resources, namely CPU, memory and storage, with the capacity of 2000 units each. We assume there are 8 types of functions in the network, and each of the physical node can provide two to four functions. The availability of each physical node is randomly distributed within [0.9, 0.99]. The network traffic along each link is carried using *Optical Orthogonal Frequency Division Multiplexing* (OOFDM), because it is a cost-effective technique to achieve Terabit-per-second transmission [20], which is needed to support the huge amount of traffic flow between data centers. Each of the links has a spectrum capacity of 24THz with a spacing of 12.5GHz per spectrum slot.

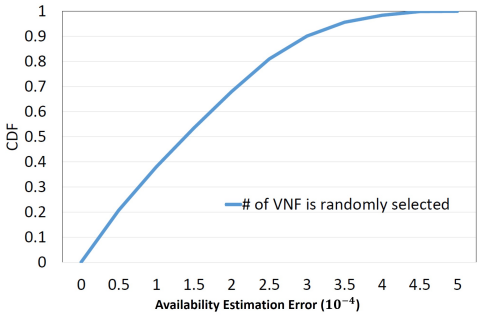


Fig. 7. CDF of the estimation error

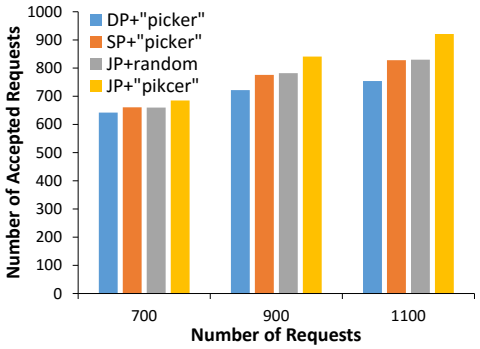


Fig. 8. Number of accepted requests w.r.t. different protection mechanism

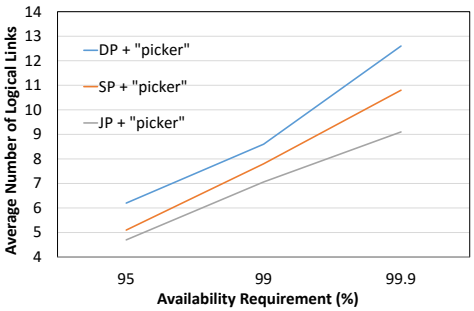


Fig. 9. Average backup VNF number w.r.t. different availability requirement

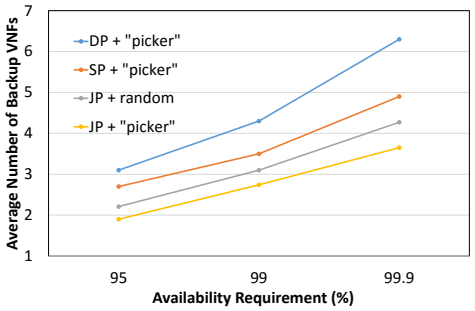


Fig. 10. Average logical links w.r.t. different availability requirement

2) SFC Requests: Each SFC request consists of two to six VNFs interconnected. Each VNF demands three types of node resources and can provide one function, and the demand for each kind of resource is uniformly distributed between 0 and 30. Each logical link has a bandwidth demand among $\{10, 40, 100, 200\}$ Gb/s with equal probability. For each SFC request, we select the availability requirement among $\{95\%, 99\%, 99.9\%\}$, similar to the ones used by Google Apps [21].

We evaluate our algorithm using a Macbook with OS X 10.9 with 1.7 GHz Intel Core i7 processor and 8GB memory. Our algorithms are implemented in C++. The statistics are the average results.

B. Availability Evaluation Accuracy

We first evaluate the availability evaluation accuracy achieved by validator proposed in Section IV.A. Table I summarizes the median estimation error when the number of VNF is randomly chosen from two to six and τ is varied.

From the table, we can see that when τ is 0.07, the estimation error is the smallest, and we fix τ to 0.07 for later simulations. With $\tau = 0.07$, we evaluate the *Cumulative Distribution Function* (CDF) of the estimation error of validator as shown in Fig. 7 when the number of request is 300 to ensure that the request acceptance ratio is 100%, and the availability threshold is set to 99.9%, as depicted in Fig. 7. We can observe that 95% of the error is smaller than 3.5×10^{-4}, which demonstrates the effectiveness of using the validator to predict the service availability.

TABLE I
MEDIAN ESTIMATION ERROR ACHIEVED BY VALIDATOR

τ	0.03	0.05	0.07	0.09	0.11
Median Error (10^{-4})	1.63	1.55	1.38	1.71	2.2

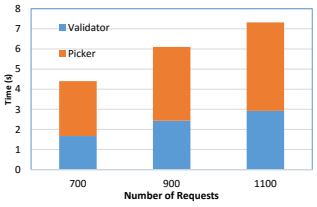


Fig. 11. Running time of picker and validator

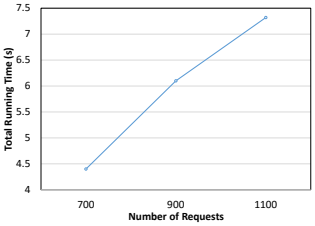


Fig. 12. Total running time

C. SFC Request Acceptance Ratio

To understand how picker and JP works, we compare the number of requests which can be accepted with different algorithms. Since a request can be accepted if and only if there is enough resource and the availability requirement can be met, the rationale behind this experiment is an algorithm with better resource efficiency can accept more request. From Fig. 8, we can see that JP + "picker" achieves the best acceptance ratio performance, and in particular, it outperforms SP and DP, both of which adopt the "picker", by 11.2% and 22.1%, respectively when the number of request is 1100. To show the effectiveness

of the "picker", we also show the case where we randomly select two VNFs for each iteration to protect as the baseline method, and JP + such random backup selection achieves the same performance as SP + "picker". Another interesting thing we can find from the figure is that when the number of requests is small (i.e., 700), all four algorithms have similar performance, while as the number of requests increases, the other three algorithms saturate faster than JP + "picker". The takeaway here is that compared with other methods, JP + "picker" can meet availability requirement while consuming less resources. Furthermore, we analyze the running time of our algorithm as shown in Fig. 11 and 12. With 1100 SFC requests, the total running time is less than 8 seconds, while validator never use more than 3 seconds.

D. Backup Resource Consumption

To further understand how JP + "picker" can save resources, we compare the average number of backup VNFs and logical links used for each SFC request w.r.t. different availability requirement. The number of request used in this experiment is 1100, and only the accepted requests are considered. As shown in Fig. 9, JP + "picker" uses 27.8% and 15.7% fewer links compared with the other two methods respectively when the availability requirement is "three nines" (i.e., 99.9%). Similar observations can be made when comparing the number of backup VNFs as illustrated in Fig. 10. We can see that JP + "picker" requires fewer number of backup VNFs. In particular, JP + "picker" can save up to 42.1% of physical nodes.

VI. Conclusion

NFV explores the virtualization technologies to offer Network-as-as-Services through connected/chained VNFs. With this new NFV technique, traditional hardware-based network appliances are replaced by network functions implemented in software that can be run on standard high-volume servers in data centers. Since telecom networks must be always on, it is critical to provide effective and efficient protection and resource allocation schemes for guaranteeing network service availability, even when catastrophic failures happen, which may bring a whole data center down. In this paper, we have proposed an online algorithm for availability-aware SFC mapping in NFV networks, which can minimize the resources allocated to SFC requests while meeting clients' heterogeneous SLA requirement. In addition we have developed a lower bound for our backup VNFs picking algorithm. Furthermore, we have shown that our design is able to evaluate service availability with a negligible estimation error in polynomial time. We have also validated our design through extensive simulations and demonstrated that it can achieve a significant performance improvement compared to traditional protection mechanisms and baseline backup VNFs picking method.

References

[1] John Donovan. How Do You Keep Pace With a 100,000 Percent Increase in Wireless Data Traffic? Software. http://about.att.com/innovationblog/3215howdoyoukeeppace

[2] Service Function Chaining Problem Statement, 2014, http://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/

[3] Service Function Chaining Use Cases In Data Centers, 2015, https://tools.ietf.org/pdf/draft-ietf-sfc-dc-use-cases-03.pdf/

[4] Network Function Virtualization White Paper, 2014, https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[5] Network Function Virtualization Architecture Framework, 2013, http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf

[6] Amazon suffers another outage, 2015, http://www.datacenterdynamics.com/it-networks/amazon-suffers-another-outage/70509.fullarticle

[7] Google Compute Engine Incident 15056, 2015, https://status.cloud.google.com/incident/compute/15056

[8] Network Functions Virtualisation (NFV) Resiliency Requirements, 2015, http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_nfv-rel001v010101p.pdf

[9] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun. Survivable virtual infrastructure mapping in a federated computing and networked system under single region failures. In *IEEE GLOBECOM*, 2010

[10] W. Yeow, C. Westphal, and U. Kozat. Designing and embedding reliable virtual infrastructure. *SIGCOMM CCR*, 2011

[11] M. Rahman, and R. Boutaba. SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization. *IEEE TNSM*, 10(2): 105-118, 2013

[12] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM*, 2011.

[13] M. Yu, Y. Yi, J. Rexford, M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM CCR*, 2008

[14] R. Potharaju, and N. Jain Demystifying the Dark Side of the Middle: A Field Study of Middlebox Failures in Datacenters. In *ACM IMC*, 2013.

[15] vSphere. https://www.vmware.com/products/vsphere/features/fault-tolerance

[16] Kwisthout, Johan. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52.9: 1452-1469, 2011

[17] M. Jerrum, and A. Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*: 482-520, 1996

[18] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba. Venice: Reliable virtual data center embedding in clouds. In *IEEE INFOCOM*, 2014

[19] Verizon Network Map. https://www22.verizon.com/wholesale/images/networkMap.png

[20] N. Cvijetic, M. Cvijetic, M. F. Huang, and E. Ip. Terabit optical access networks based on WDM-OFDMA-PON. *IEEE/OSA JLT*, 30(4): 493-503, 2012

[21] Google Apps Service Level Agreement. http://www.google.com/apps/intl/en/terms/sla.html

[22] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. *IEEE FOCS*, 2011

[23] T. Cormen. Introduction to algorithms. MIT press, 2009.

[24] L. Gong, Y. Wen, Z. Zhu, and T. Lee. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *IEEE INFOCOM*, 2014.

[25] F. Hao, M. Kodialam, T.V. Lakshman, and S. Mukherjee. Online allocation of virtual machines in a distributed cloud In *IEEE INFOCOM*, 2014.

[26] D. Xie, N. Ding, Y.C. Hu, and R. Kompella. The only constant is change: incorporating time-varying network reservations in data centers. *SIGCOMM Comput. Commun. Rev.*, 42(4): 199-210, 2012

Appendix

We first formulate the availability-aware SFC mapping problem as a Boolean formula. For one physical node n, p_n represents if node n is used for mapping for the request, and q_n and $\bar{q_n}$ denote if node n is used as a mapping node for one VNF in a virtual request or a backup VNF, respectively. Assume this physical node n can provide a set of functions $\{f_i | f_i \in F_n\}$, then we use $\{x_n^i | f_i \in P_n\}$ to show which one or

two functions node n provides. For example, assume a PS has 3 PNs which n_1, n_2 and n_3 can provide functions f_1, f_2, f_3, f_2 and f_1, f_3 and given a SFC request that needs functions f_2, f_3, then the Boolean formula can be written as

$$\begin{aligned}\varphi &= C(n_1) \wedge C(n_2) \wedge C(n_3) \wedge eval(n_1, n_2, n_3) \\ &= (((p_1 \wedge q_1) \wedge (x_1^1 \wedge \bar{x_1^2} \wedge \bar{x_1^3}) \wedge (\bar{x_1^1} \wedge x_1^2 \wedge \bar{x_1^3}) \\ &\quad \wedge (\bar{x_1^1} \wedge \bar{x_1^2} \wedge x_1^3)) \vee ((p_1 \wedge \bar{q_1}) \wedge (x_1^1 \wedge x_1^2 \wedge \bar{x_1^3}) \\ &\quad \wedge (\bar{x_1^1} \wedge x_1^2 \wedge x_1^3) \wedge (x_1^1 \wedge \bar{x_1^2} \wedge x_1^3)) \vee (\bar{p_1} \wedge \bar{x_1^1} \wedge \bar{x_1^2} \wedge \bar{x_1^3})) \\ &\quad \wedge ((p_2 \wedge x_2^2) \vee (\bar{p_2} \wedge \bar{x_2^2})) \wedge (((p_3 \wedge q_3) \wedge (x_3^1 \wedge \bar{x_3^3}) \\ &\quad \wedge (\bar{x_3^1} \wedge x_3^3)) \vee ((p_3 \wedge \bar{q_3}) \wedge (x_3^1 \wedge x_3^3)) \vee (\bar{p_3} \wedge \bar{x_3^1} \wedge \bar{x_3^3})) \\ &\quad \wedge (((x_1^2 \wedge x_1) \vee (x_2^2 \wedge x_2)) \wedge ((x_1^3 \wedge x_1) \vee (x_3^3 \wedge x_3)))\end{aligned}$$

where $C(n_i)$ shows the constraints for node i, and $eval(n_1, n_2, n_3)$ presents which nodes can provide which functions. x_1, x_2 and x_3 is 1 if this physical node can function normally at a given time and 0 otherwise. We are trying to find the minimum number of nodes needed to be selected, which is equivalent to finding the maximum set of nodes A whose value of p_n can be set to 0, and let the rest of the nodes satisfy the Boolean formula with probability greater or equal to certain threshold. As the first step, we show that verifying if the availability is above clients' requirement optimally is not a viable option.

Proof for Theorem 1. By the definition of language in PP, it is clear that VA problem is in PP. Note that MAJSAT [16] is a PP-complete problem. To show PP-completeness, we can reduce MAJSAT problem to VA problem. Note that, for an instance ϕ with n variables of MAJSAT, the number of all possible assignments to ϕ is 2^n. Thus, we have

$$\begin{aligned}\phi \in MAJSAT &\iff \text{the number of assignments that} \\ &\qquad \text{satisfies } \phi \text{ is greater than } 2^{n-1} \\ &\iff \Pr[\phi(x)] > \frac{1}{2} \text{ with } x \in \{0,1\}^n\end{aligned}$$

Given that VA problem's instance is a pair (ϕ, θ) consisting of a Boolean formula ϕ and a threshold θ. Hence, with a MAJSAT instance ϕ, we can set instance $(\phi, 1/2)$ for VA problem. To verify the correctness,

$$\begin{aligned}\phi \in MAJSAT &\iff \Pr[\phi(x)] > \frac{1}{2} \text{ with } x \in \{0,1\}^n \\ &\iff \text{the probability that a given formula} \\ &\qquad \phi \text{ can be satisfiedis greate than a} \\ &\qquad \text{given threshold } \frac{1}{2} \\ &\iff (\phi, \frac{1}{2}) \in VR,\end{aligned}$$

$$\begin{aligned}\phi \notin MAJSAT &\iff \Pr[\phi(x)] \leq \frac{1}{2} \text{ with } x \in \{0,1\}^n \\ &\iff \text{the probability that a given formula} \\ &\qquad \phi \text{ can be satisfieis NOT greater than} \\ &\qquad \text{a given threshold } \frac{1}{2} \\ &\iff (\phi, \frac{1}{2}) \notin VA.\end{aligned}$$

Thus, it is a valid many-one reduction from MAJSAT problem to VA problem. Therefore, VA problem is also PP-complete.

Proof for Theorem 2. We can construct a nondeterministic oracle Turing machine N with oracle that solves VR problem, which conducts the following three steps to solve the given instance (Y, ϕ, θ) of DE problem, where Y is a set of variables:

1) Randomly guess a solution to set Y, which takes time $O(|Y|)$
2) Hardcode the guess to ϕ to obtain ϕ', which takes time $O(|\phi|)$
3) Query the VA oracle with (ϕ, θ)

Since $O(|Y|)$ and $O(|\phi|)$ are polynomials in terms of n, this satisfies the definition of NP^{PP} class. Hence, DE problem is in NP^{PP}. To show NP^{PP}Ccompleteness, we reduce E-MAJSAT [16] problem to DE problem. In E-MAJSAT problem, for an instance (k, ϕ), (we represent a sequence of variables x as $x_1 x_2 \cdots x_n$.)

$$\begin{aligned}(k, \phi) \in E - MAJSAT &\iff (\exists x_1 x_2 \cdots x_k \in \{0,1\}^k) \\ &\qquad \sharp(\text{assignments to } x_{k+1} \cdots x_n \\ &\qquad \text{that satisfies } \phi) > 2^{n-k} \\ &\iff (\exists x_1 x_2 \cdots x_k \in \{0,1\}^k) \\ &\qquad \Pr[\phi(x) | x_1 x_2 \cdots x_k] > \frac{1}{2},\end{aligned}$$

where $\sharp(A)$ denotes the number of elements in set A. With such an E-MAJSAT instance, we first define a set of variables as $Y = x_1, x_2, \cdots, x_k$ and then set instance $(Y, \phi, 1/2)$ for DE problem. To verify correctness

$$\begin{aligned}(k, \phi) \in E - MAJSAT &\iff (\exists x_1 x_2 \cdots x_k \in \{0,1\}^k) \\ &\qquad \Pr[\phi(x) | x_1 x_2 \cdots x_k] > \frac{1}{2} \\ &\iff \text{there exists a solution to set} \\ &\qquad \text{Y such that the probability} \\ &\qquad \text{that a given } \phi \text{ can be satisfied} \\ &\qquad \text{is greater than a given} \\ &\qquad \text{threshold } \frac{1}{2} \\ &\iff (Y, \phi, \frac{1}{2}) \in DE,\end{aligned}$$

$$
\begin{aligned}
(k,\phi) \notin E-MAJSAT &\Longleftrightarrow (x_1x_2\cdots x_k \in \{0,1\}^k) \\
&\quad \Pr[\phi(x)|x_1x_2\cdots x_k] > \frac{1}{2} \\
&\Longleftrightarrow (\forall x_1x_2\cdots x_k \in \{0,1\}^k) \\
&\quad \Pr[\phi(x)|x_1x_2\cdots x_k] \leq \frac{1}{2} \\
&\Longleftrightarrow \text{there DOESNOT exist a} \\
&\quad \text{solution to set Y such that} \\
&\quad \text{the probability that a given} \\
&\quad \phi \text{ can be satisfied is greater} \\
&\quad \text{than a given threshold} \\
&\Longleftrightarrow (Y,\phi,\frac{1}{2}) \notin DE,
\end{aligned}
$$

Thus, this is a valid many-one reduction from E-MAJSAT problem to DE problem. Therefore, DE problem is NP^{PP}-complete.

Proof for Theorem 3. To show co-NP$^{\text{PP}}$-completeness, we can construct a many-one reduction from A-MAJSAT problem to this problem. Concretely, note that the formula ϕ in the input instance (k,ϕ) is not necessarily of monotone form, while the Boolean formula in instance for problem LM is monotone. Hence, in order to transfer an A-MAJSAT instance to a LM instance, we need to employ the standard way of converting general Boolean formula to monotone CNF Boolean formula. Observe that, if fixing a subset of variables Y, in such monotone CNF Boolean formula, the number of satisfying assignments would become a minimum with all variables in Y set to be 0. With this standard way, it is also easy to derive such a set Y from k in A-MAJSAT instance. At this point, we set a threshold θ as $\frac{1}{2}$ and then get an instance $(Y,\phi',\frac{1}{2})$. Then, we can use an oracle solving problem LM to solve A-MAJSAT problem. It is easy to check the validity of this many-one reduction. Since A-MAJSAT problem is co-NP$^{\text{PP}}$-complete, therefore, problem LM is also co-NP$^{\text{PP}}$-complete.

Proof for Theorem 4. By the definition of $NP^{NP^{PP}}$ [16], we can construct a polynomial-time bounded nondeterministic oracle Turing machine M to accept our problem with oracle to a problem in $co-NP^{PP}$ as follows:
Taken an instance (ϕ,θ) of our problem as input

1) Randomly guess a maximum set A that will meet our property
2) Use (A,ϕ,θ) to access the LM oracle that decides maximal set

Clearly, before verifying, the randomly generated set A by machine M is just a possible candidate for maximum set. To make sure that, M uses LM oracle to verify the correctness. Note that we reply on the power of nondeterministic Turing machine to guess a possible solution, which takes $O(n)$ time. Also LM is $co-NP^{PP}$-complete. Therefore, our problem is in $NP^{NP^{PP}}$.