SAM: Maximizing Service Function Chain Availability in Cloud Data Centers

Sterling Abrahams*, Bin Tang*, Deng Pan†
*Department of Computer Science, California State University Dominguez Hills
sabrahams1@toromail.csudh.edu, btang@csudh.edu
†School of Computing and Information Sciences, Florida International University, pand@fiu.edu

***Abstract*—Service function chaining (SFC), consisting of a sequence of virtual network functions (VNFs), provides effective and flexible network service management in a cloud computing environment. Due to the vulnerabilities of software-implemented VNFs, existing research has introduced VNF backup servers to achieve the fault-tolerance of VNFs and improve the availability of SFCs. However, they either do not consider the failures of backup servers or do not aim to maximize the availability of the entire SFC. In this paper, we study how to maximize the availability of an SFC, considering that both VNFs and backup servers can fail. We refer to the problem as SAM: service function chaining availability maximization problem. Given an SFC and a set of backup servers placed inside a cloud data center network, the failure probabilities of the VNFs and the backup servers, the goal of SAM is to assign backup servers to VNFs to maximize the availability of the SFC while satisfying the backup capacity constraint of the servers. We design a suite of optimal and efficient algorithms to solve SAM. Via extensive simulations with different network parameters, we show that our work outperforms the existing research by up to 21.7% in SFC availability, demonstrating the effectiveness of our algorithms in achieving high fault tolerance of SFC in cloud data centers.**

Keywords – Virtual Network Functions, Service Function Chaining, Availability, Cloud Data Centers

I. Introduction

Motivations. Network Function Virtualization (NFV) is a cost-effective and flexible technique to provision network services in a cloud environment [11], [19]. With NFV, proprietary hardware middleboxes (MBs) such as firewalls, cache proxies, and load balancers can now be implemented as virtual network functions (VNFs) running as lightweight containers on general-purpose commodity hardware [5], [18]. Being instantiated and deployed dynamically, VNFs provide performance and security guarantees to cloud user applications flexibly and cost-effectively. In particular, cloud providers create *service function chains* (SFCs), which consist of a sequence of VNFs of different functions, and require virtual machine (VM) application traffic to traverse the SFC to receive the performance and security support [9].

Despite the achieved flexibility and low cost, implementing hardware MBs into software VNFs poses significant challenges to achieving the reliability and fault-tolerance of VNFs. Compared to dedicated proprietary hardware MB devices, the VNFs are more vulnerable to software misconfiguration, bugs, and malfunctions than their hardware counterpart [16]. Besides, as the VM traffic must traverse all the VNFs in the SFC to achieve requested security and performance guarantees, the failure of any of its constituent VNFs renders the entire SFC inoperative and its service unusable, further exacerbating the problem. Therefore, achieving the fault-tolerance for VNFs and maximizing the availability of the SFC becomes a critical problem.

One common approach is to deploy *VNF backup servers* [13], [14], [6], [7], where a backup copy of the VNF is installed.[1] For each active VNF in the SFC, its state information for failure recovery is synchronically updated to its assigned backup server. If any active VNF fails, its corresponding backup VNF is activated, and the VM traffic can then be redirected immediately to it to finish the remaining SFC traversal. This approach has been shown to increase the fault-tolerant capabilities of the SFCs dramatically [20], [24], [21], [4].

However, we have two observations about all the above research. First, as VNFs are usually more vulnerable than their backup servers, much of the current research only considered the failures of the VNFs [13], [14], [6] or the devices that host them [7], [20], [10], and did not consider the failures of the backup servers. In practical NFV environments, backup servers are implemented as virtual machines or containers [10], thus having non-negligible failure rates as well. Therefore, to guarantee a robust availability provision for SFCs, it is critical to consider the failures of the backup servers. Second, as backup servers are resources for redundancy, most of the research focused on reducing such resource consumption (e.g., number of backup servers) while achieving the required high availability of the SFCs [6], [7], [24]. This implicitly assumes that there are always enough backup resources to achieve such high availability. This, unfortunately, is not always the case for cloud providers, as there are always a limited amount of resources at their disposal.

Our Contributions. In this paper, we ask the following question: *Considering the failures of both the VNFs and their backup servers (with limited backup capabilities) in a cloud environment, how to allocate the backup resources to VNFs to maximize the availability of the entire SFC?*

We identify, formulate, and solve a new algorithmic problem called SAM: service function chaining availability

[1]Note that a backup server is not necessarily a physical server. It could be a virtual machine (VM) or a dedicated set of libraries.

maximization. Given an SFC consisting of a sequence of VNFs, a set of backup servers with limited backup resource capacities, and the failure probabilities of both VNFs and backup servers, the goal of SAM is to assign backup servers to VNFs to maximize the availability of the SFC while satisfying the resource capacities of the backup servers. SAM can be modeled as a minimum cost flow problem [2], which can be solved efficiently and optimally. We also propose two greedy heuristic algorithms to solve SAM, one drawing inspiration from existing research [10]. Via simulations of different network parameters, we show that our algorithms outperform the existing research [10] that considers the failures of both VNFs and backup servers by up to 21.7%. This demonstrates the effectiveness of our algorithm in achieving the high availability of SFC in cloud environments.

II. Related Work

There are two mechanisms for VNF backup servers. One is called *dedicated backup*, wherein at least one standby VNF instance (on an exclusive backup server) is required for each active VNF. Fan. et al. [6] studied how to minimize the number of backup servers while meeting the overall availability demand of the SFC. Their follow-up work [7] presented a more general framework to provision SFC request availability with multiple layers of connected devices and heterogeneous failure processes. The other is called *shared backup*, wherein a backup server is allowed to protect multiple VNFs. Kanizo et al. [13] considered how to assign backup servers to MBs to maximize the overall probability of recovering all failing VNFs or maximize the number of functions that can be recovered simultaneously. They provided a few heuristic algorithms. Their following work [14] analyzed the maximum number of failing VNFs that can be fully recovered by a given number of backup servers. It also showed the minimum number of backup servers required to guarantee full recovery for a given number of failing functions.

We take the second approach as it reduces costs by resource-sharing among the standby instances of the different VNFs. Besides, all the above works only considered the failure probability of VNFs while we consider the failures of both VNFs and backup servers in a more practical cloud environment.

There are only a few works that considered the failure rates of both the VNFs and their backup servers. Zhang et al. [24] considered that VNF may request heterogeneous resources and proposed to minimize the backup resource consumption while meeting the overall availability demand. He et al. [10] aimed to find an assignment of backup servers to VNFs to minimize the maximum unavailability among all the active VNFs. They showed the problem is NP-hard and proposed a few heuristic algorithms. Although their goal improves the availability of the most vulnerable VNF in an SFC, it does not necessarily improve the availability of the entire SFC. As the cloud traffic must traverse the entire SFC to receive its network services, optimizing the availability of the entire SFC is more relevant to the cloud operators to provide quality of service to cloud users. Via extensive simulations, we show that our work indeed achieves higher SFC availability than that of [10].

Qu et al. [20] proposed a VNF placement and traffic routing optimization framework that jointly maximizes the availability of SFCs and minimizes the end-to-end delays of service requests. Huang et al. [12] aimed to maximize the number of requests admitted while meeting the specified reliability requirement of each admitted request. This work was then extended to mobile edge cloud with IoT applications, to study the VNF service reliability by jointly considering the reliability of both VNF instances and cloudlets [17]. Wang et al. [22] focused on the parallelized SFC placement problem in DCN considering availability guarantee and resource optimization. Shang et al. [21] proposed a VNF reliability-aware adaptive deployment scheme named RAD to efficiently place and back up SFCs over both the edge and the cloud. RAD does not assume failure rates of VNFs but instead strives to find the sweet spot between the desired availability of SFCs and the backup cost. Kong et al. [15] adopted both backup network paths and VNF replicas to guarantee an SFC's availability.

However, none of the above works studied the problem of SFC availability maximization, which is the topic of this paper. To the best of our knowledge, our work is the first to explicitly maximize the availability of the entire SFC under the constraints of limited backup resources while considering the failure of both VNFs and backup servers.

III. Problem Formulation of SAM

Problem Formulation. We model a VNF-enabled data center as an undirected general graph $G(V = V_p \cup V_s, E)$, where $V_s = \{s_1, s_2, ..., s_{|V_s|}\}$ is the set of $|V_s|$ switches and $V_p = \{p_1, p_2, ..., p_{|V_p|}\}$ is the set of $|V_p|$ physical machines (PMs), and E is the set of edges connecting two nodes (either a switch or a PM) in V. Each switch is attached to a *VNF backup server* which can install multiple backup VNFs. Each backup server $i \in V_s$ has a *resource capacity* of r_i, indicating that it has the source to install and keep constant updates with at most r_i VNFs. Besides, backup server i has a failure probability p_i. We use V_s to refer to both the set of switches and the set of backup servers, as the context is clear to distinguish them.

There is a service function chain (SFC) in G consisting of m VNFs, which is denoted by $M = \{v_1, v_2, ..., v_m\}$. $v_j \in M$ is installed at switch $s(j)$. The failure probability of v_j is q_j. We assume that both p_i and q_j are independent and identically distributed random variables. To recover from the possible failure, each VNF must be assigned a backup server and constantly update with it. If VNF v_j's assigned backup server is s_i, then the *unavailability* of v_j, denoted by $f_{j,i} = q_j \cdot p_i$, is the probability of both v_j and s_i fail simultaneously. Thus the *availability* of v_j, which is the probability that v_j is available, is $1 - q_j \cdot p_i$. We define a *backup server assignment function* as $a : [1, 2, ..., m] \rightarrow [1, 2, ..., |V_s|]$, indicating that the backup server of VNF $v_j \in M$ is $s_{a(j)} \in V_s$. Then *the availability of the entire SFC* is defined as follows.

Definition 1: (**Availability of an SFC.**) The availability of an SFC M is defined as the probability that all the VNFs

in M are available. Let $\mathcal{A}(a)$ denote the *availability of the SFC* given a backup server assignment function a; $\mathcal{A}(a) = (1 - f_{1,a(1)}) \cdot (1 - f_{2,a(2)}) \cdot ... \cdot (1 - f_{m,a(m)})$. □

The objective of SAM is to find an a to maximize $\mathcal{A}(a)$ under the constraint that each VNF server $i \in V_s$ can be the backup server of at most r_i VNFs; that is, $|\{j | i \in s(j), 1 \leq j \leq m\}| \leq r_i$, $\forall i \in V_s$. Table I shows all the notation.

Theorem 1: Maximizing $\mathcal{A}(a)$ is equivalent to minimizing $\sum_{1 \leq j \leq m} f'(j, a(j))$, where $f'(j, a(j)) = \log \frac{1}{(1 - f_{j,a(j)})}$.

Proof: To maximize $\mathcal{A}(a)$ is to maximize $\log \mathcal{A}(a)$, where

$$\begin{aligned} \log \mathcal{A}(a) &= \log\big((1 - f_{1,a(1)}) \cdot ... \cdot (1 - f_{m,a(m)})\big) \\ &= \log(1 - f_{1,a(1)}) + ... + \log(1 - f_{m,a(m)}). \end{aligned} \tag{1}$$

Next, maximizing $\log \mathcal{A}(a)$ is equivalent to minimizing $-\log \mathcal{A}(a)$, where

$$-\log \mathcal{A}(a) = \log \frac{1}{(1 - f_{1,a(1)})} + ... + \log \frac{1}{(1 - f_{m,a(m)})}. \tag{2}$$

Let $f'(j, a(j)) = \log \frac{1}{(1 - f_{j,a(j)})}$. Therefore the goal of the SAM is to minimize $\sum_{1 \leq j \leq m} f'(j, a(j))$. ∎

Discussions. Note that in our problem formulation, multiple VNFs of the same SFC can have the same backup server as long as its resource capacity allows. However, in this case, the failure among multiple VNFs in the same backup server can still be considered independent. This is because the software failure of VNFs and the hardware failure of the backup are not correlated. In this work, we only consider a single SFC in order to achieve an optimal solution and rigorous analysis. It is non-trivial to extend our problem and solution to the case of multiple SFCs. We leave it as our future work.

IV. Algorithmic Solutions to SAM

In this section, we present an optimal and efficient algorithm to solve SAM based upon Theorem 1. In particular, we show that minimizing $\sum_{1 \leq j \leq m} f'(j, a(j))$ on data center graph $G(V, E)$ is equivalent to a minimum cost flow (MCF) problem on a flow network $G(V', E')$ properly transformed from $G(V, E)$. We first introduce the MCF problem and its algorithms and then show the detailed procedures of the transformation. We also present two heuristic greedy algorithms, one of which is inspired by an existing work [10].

MCF Problem and Algorithms. Let $G' = (V', E')$ be a directed graph representing a flow network, wherein the capacity and cost of an edge $(u, v) \in E'$ are denoted by $p(u, v)$ and $c(u, v)$, respectively. The amount of supply from *source node* $s \in V'$ and the amount of demand at *sink node* $t \in V'$ is d. Denote a flow on edge (u, v) as $f(u, v)$, $f : E' \to \mathbb{R}^+$. $f(u, v)$ is subject to (a) *capacity constrain*t: $f(u, v) \leq p(u, v), \forall (u, v) \in E'$ and (b) *flow conservation constraint*: $\sum_{u \in V'} f(u, v) = \sum_{u \in V'} f(v, u)$, for each $v \in V' \backslash \{s, t\}$. The goal of MCF is to find a flow function f such that the total cost of sending d amount of flow from s to t, which is $\Sigma_{(u,v) \in E'} \big(c(u, v) \cdot f(u, v)\big)$, is minimized.

MCF can be solved optimally and efficiently by many combinatorial algorithms including cycle-canceling, successive shortest path, out-of-kilter algorithm, cost- and capacity-scaling, and network simplex algorithm [2]. In this paper, we adopt the scaling push-relabel algorithm proposed by Goldberg [8], which is by far the MCF algorithm implementation that has the highest performance. Its time complexity is $O(A^2 \cdot B \cdot \log(A \cdot C))$, where A, B, and C are the number of nodes, number of edges, and maximum edge capacity in the flow network, respectively.

Graph Transformation. Next, we transform the data center graph $G(V, E)$ into a flow network $G'(V', E')$. Fig. 1 shows the following five steps.

Step I. $V' = \{s\} \cup \{t\} \cup M \cup V_s$. Here, s is the source node and t is the sink node in the flow network, $M = \{v_1, v_2, ..., v_m\}$ is the set of m VNFs, and V_s is the set of backup servers.

Step II. $E' = \{(s, v_j)\} \cup \{(v_j, s_i)\} \cup \{(s_i, t)\}$, where $v_j \in M$ and $s_i \in V_s$. Note that it is a complete bipartite graph between M and V_s.

Step III. For each edge (s, v_j), set its capacity as 1 and cost as 0. For each edge (s_i, t), set its capacity as r_i, the resource capacity of s_i, and cost as 0.

Step IV. For each edge (v_j, s_i), set its capacity as 1 and cost as $f'(j, i)$.

Step V. Set the supply at s and the demand at t as m, the number of VNFs in the SFC M.

Thus, $|V'| = m + |V_s| + 2$ and $|E'| = m + |V_s| + m \cdot |V_s|$.

TABLE I
NOTATION SUMMARY

Notation	Explanation
$G(V, E)$	Data center network, $V = V_p \cup V_s$
V_p	Set of $\|V_p\|$ PMs in $G(V, E)$
V_s	Set of $\|V_s\|$ switches or VNF backup servers in $G(V, E)$
M	An SFC of m VNFs v_j, $1 \leq j \leq m$
i	index for VNF backup servers, $1 \leq i \leq \|V_s\|$
j	index for VNFs, $1 \leq j \leq m$
r_i	The resource capacity of VNF backup server $s_i \in V_s$
p_i	Failure probability of VNF backup server $s_i \in V_s$
q_j	Failure probability of VNF $v_j \in M$
$f_{j,i}$	Unavailability of VNF v_j with backup server $s_i \in V_s$
$s(j)$	The switch where VNF v_j is installed
a	Backup server assignment function
$s_{a(j)}$	Backup server $s_{a(j)} \in V_s$ of VNF v_j under assignment a
$\mathcal{A}(a)$	SFC availability under backup server assignment a

Theorem 2: The SAM in data center network $G(V, E)$ is equivalent to the MCF in flow network $G'(V', E')$.

Proof: We need to show that by applying the MCF algorithm upon $G'(V', E')$, it achieves the maximum availability for the SFC M in $G(V, E)$ while satisfying the capacity constraints of backup servers and that each VNF is assigned with one backup server.

First, with the above transformation, sending m amount of flow from s to t in $G'(V', E')$ ensures that each of the m VNFs in M is assigned to one backup server. In particular, since the amount of supply at s is m and the capacity of each edge (s, v_j) is one, a valid flow of m amount from s to t must have one amount of flow on edge (s, v_j). Then, due to the flow

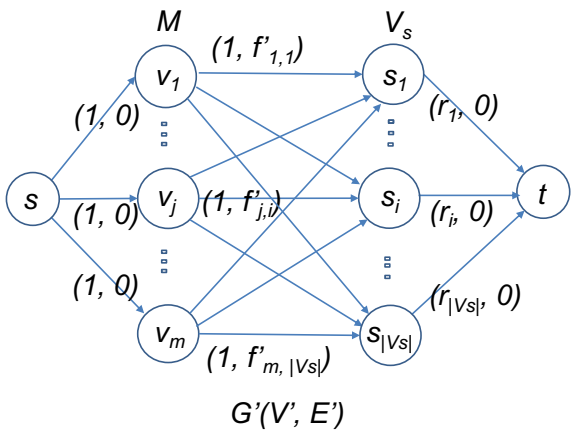


Fig. 1. The SAM is equivalent to the MCF problem. The values in parenthesis are the capacity and cost of an edge, respectively.

conservation on each node v_j, there must be one amount of flow that comes out of v_j and goes to a backup server s_j. This indicates that each v_j will be assigned to one of the backup servers at the end of MCF computation.

Next, the edge capacity of r_i on edge (s_i, t) ensures that no more than r_i amounts of flow coming out of backup server s_i. The flow conservation at s_i thus ensures that no more than r_i amounts of flow come into s_i. This indicates that s_i can be the backup server of at most r_i VNFs, satisfying the capacity constraint of s_i.

Finally, as the cost on (v_j, s_i) is $f'_{j,i}$, the MCF algorithm finds a backup server to VNF assignment with a minimum sum of the costs of $f'_{j,i}$ on all edges. Combined with Theorem 1, this shows that this assignment indeed achieves maximum availability for SFC M. ∎

Time Complexity. Our optimal algorithm that finds the maximum SFC availability consists of two steps: the graph transformation from $G(V, E)$ to $G'(V', E')$ and then applying scaling push-relabel MCF algorithm [8] on $G'(V', E')$. The graph transformation takes $O(m \cdot |V_s|)$ and the MCF algorithm takes $O\big(|V'|^2 \cdot |E'| \cdot \log(|V'| \cdot \max\{r_i\})\big)$. As $|V'| = m + |V_s| + 2$, $|E'| = m + |V_s| + m \cdot |V_s|$, the time complexity of our algorithm is $O\big(|V_s|^3 \cdot m \cdot \log(|V_s| \cdot \max\{r_i\})\big)$.

Heuristic Algorithms for SAM. In addition, we propose two more time-efficient heuristic algorithms. One is called *Sorted Greedy Algorithm* (i.e., Algo. 1) and the other is called *Reverse Sorted Greedy Algorithm* (i.e., Algo. 2).

Sorted Greedy Algorithm. Algo. 1 works as follows. We first sort all the VNF backup servers in the non-descending order of their failure probabilities (line 1) and sort all the VNFs in the non-ascending order of their failure probabilities (line 2). Then for each backup server i, we assign r_i number of VNFs that have not been assigned to any backup server (lines 5-11). This continues until all the m VNFs are assigned their backup servers. The time complexity of Algo. 1 is $O(|V_s| \cdot \log|V_s| + m \cdot \log m)$.

Algorithm 1: Sorted Greedy Algorithm for SAM.
Input: Data center $G(V, E)$ with failure probability p_i for backup server s_i and failure probability q_j for VNF v_j;
Output: VNF backup server assignment a and SFC availability $\mathcal{A}(a)$.

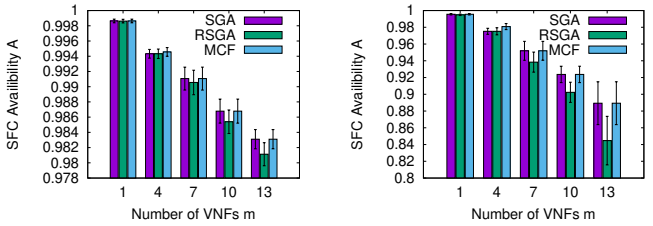
0. **Notations**:
   l: index of the resources for VNF backup servers;
   $flag$: assignment is done or not, initially false;
1. Sort all the backup servers in non-descending order of q_i; WLOG, $q_1 \leq q_2 \leq ... \leq q_{|V_s|}$;
2. Sort all the VNFs in non-ascending order of p_j; WLOG, $p_1 \geq p_2 \geq ... \geq p_m$;
3. $l = 1$;
4. **for** $(i = 1;\ i \leq |V_s|;\ i{+}{+})$
5. **for** $(j = l;\ j < l + r_i;\ j{+}{+})$
6. $a(j) = i$; // Assign server s_i to v_j
7. **if** $(j \geq m)$
8. $flag = true$;
9. **break**;
10. **end if;**
11. **end for;**
12. **if** $(flag == true)$ **break**;
13. **end if;**
14. $l = l + r_i$; // Assign next backup server;
15. **end for;**
16. **RETURN** a and $\mathcal{A}(a)$.

Reverse Sorted Greedy Algorithm. Algo. 2 is different from the Sorted Greedy Algorithm only in one step. Instead of sorting all the VNFs v_j in the non-ascending order of p_j (Algo. 1, line 2), it sorts them in the non-descending order of p_j. The rest of the algorithm is the same as lines 3-16 in Algo. 1 and thus are omitted below. Its time complexity is also $O(|V_s| \cdot \log|V_s| + m \cdot \log m)$.

Algorithm 2: Reverse Sorted Greedy Algorithm for SAM.
Input: Data center $G(V, E)$ with failure probability p_i for backup server s_i and failure probability q_j for VNF v_j;
Output: VNF backup server assignment a and SFC availability $\mathcal{A}(a)$.

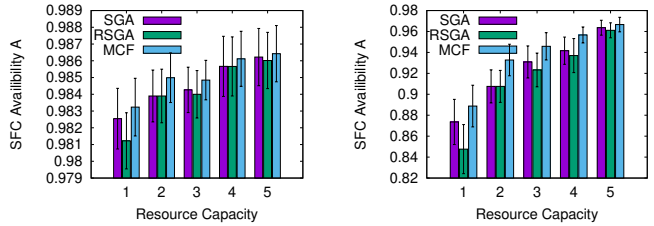
0. **Notations**:
   l: index of the resources for VNF backup servers;
   $flag$: assignment is done or not, initially false;
1. Sort all the backup servers in non-descending order of q_i; WLOG, $q_1 \leq q_2 \leq ... \leq q_{|V_s|}$;
2. Sort all the VNFs in non-descending order of p_j; WLOG, $p_1 \leq p_2 \leq ... \leq p_m$;

Discussions and State-of-the-Art [10]. The rationale underlying both algorithms is that to maximize the availability of the entire SFC, we should first utilize backup servers with the lowest failure probabilities. However, it is not clear if VNFs with the highest (Algo. 1) or lowest (Algo. 2) failure probabilities should be assigned first, which is explored in the simulations. He et al. [10] is one of the few works



(a) $0.01 \leq p_i \leq 0.05$. (b) $0.01 \leq p_i \leq 1$.

Fig. 2. Varying m, number of VNFs. Here, $r_i = 1$, $0.025 \leq q_j \leq 0.175$, $k = 8$.



(a) $0.01 \leq p_i \leq 0.05$. (b) $0.01 \leq p_i \leq 1$.

Fig. 3. Varying r_i, resource capacity of backup servers. Here, $0.025 \leq q_j \leq 0.175$, $m = 13$, $k = 8$.

that considered both VNFs and backup server failures. They studied how to assign backup servers for MBs to minimize the maximum unavailability of the MBs. As the problem is NP-hard, they proposed a few heuristic algorithms, one of them is the Sorted Greedy Algorithm (i.e., Algo. 1). In Section V, we show that our MCF algorithm constantly achieves higher SFC availability than SGA under different network parameters.

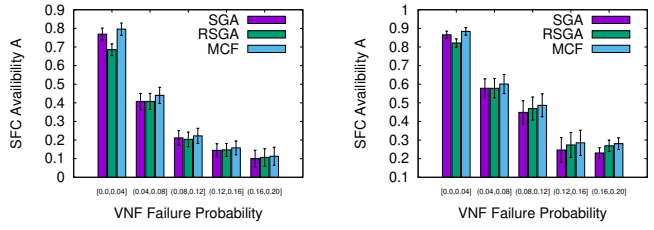
V. Performance Evaluation

In this section, we compare our designed algorithms with the existing work [10]. We refer to the optimal MCF algorithm as **MCF**, the Sorted Greedy Algorithm (i.e.. Algo. 1) as **SGA**, and the Reverse Sorted Greedy Algorithm (Algo. 2) as **RSGA**.

Simulation Setup. For the simulations, we use k-ary fat-trees [3], where k is the number of ports of each switch. Fat-tree topologies are widely adopted in data centers to interconnect commodity Ethernet switches.[2] We consider both $k = 8$ and $k = 12$ fat-trees. Following [10], the failure probability of VNFs is randomly in the range of [0.025, 0.175], and the failure probability of backup servers is in the range of [0.01, 0.05] or [0.01, 1], unless otherwise mentioned. Each data point in the plot is an average of 10 runs with a confidence interval of 95%. For a fair comparison, we apply different algorithms on the same SAM instance with the same VNF placement and failure probabilities of VNF and backup servers. As real-world SFCs [1] consist of up to 13 VNFs, we consider the number of VNFs m in an SFC to be at most 13 unless otherwise mentioned. The VNFs are randomly installed on the different switches in the data center. The backup servers are attached to the switches that do not have VNFs installed.

The Case of $r_i = 1$. We first study a basic case of r_i wherein each backup server's backup capacity is 1; that is, it can only install one backup VNF. Fig. 2 compares different algorithms by varying numbers of VNFs m while setting the failure probabilities of VNFs q_j randomly in $[0.025, 0.175]$. Fig. 2(a) focuses on more reliable backup servers with $p_i \in [0.01, 0.05]$ while Fig. 2(b) on less reliable backup servers with $p_i \in [0.01, 1]$. For both cases, we observe that with the increase of m, the SFC availability for each algorithm decreases. This shows the fundamental vulnerability of SFC: the more VNFs it has, the more it is prone to failure. Second, we observe that MCF achieves the same SFC availability as the SGA does in most cases, and both perform better than RSGA. This partially supports our claim that the MCF algorithm is optimal. That SGA performs better than RSGA shows that when $r_i = 1$, assigning the most vulnerable VNFs (i.e., with the highest failure probabilities) to the most reliable backup server (i.e., with the smallest failure probability) is a good practice to increase the availability of the entire SFC chain. Finally, comparing Fig. 2(a) and (b) shows that for each algorithm, the resultant SFC availability is lower when backup servers have higher failure rates.

The Effects of Varying r_i. Next, Fig. 3 investigates the effects of varying r_i. First, it shows that with the increase of r_i, the resulting SFC availability from all the algorithms increases. This is because with larger r_i, backup servers with low failure probabilities are able to accommodate and back up more VNFs, improving the SFC availability. Second, unlike Fig. 2, which shows that MCF and SGA perform mostly the same at $r_i = 1$, it shows that MCF outperforms SGA in most cases. When backup servers have more resources, the MCF, being optimal, is more effective than SGA in achieving SFC availability. Finally, the SFC availability in Fig. 3(b) is at least 0.981, which is much higher than those between 0.84 to 0.96 in Fig. 3(a). This again demonstrates that SFC availability gets lower when backup servers have higher failure rates.



(a) $r_i = 5$. (b) $r_i = 10$.

Fig. 4. Varying q_j, VNF failure probability. Here, $0.01 \leq p_i \leq 1$, $m = 100$, $k = 12$.

The Effects of Varying q_j. Finally, we investigate the effects of varying q_j, the VNF failure probabilities, in large-scale $k = 12$ fat-tree cloud data centers. Following [10], we consider SFCs of $m = 100$ VNFs.[3] Fig. 4(a) and (b) show the performance comparison for $r_i = 5$ and $r_i = 10$, respectively. Again, we observe MCF performs the best in all the cases with larger SFC availability, partially supporting our claim

[2]However, our designed algorithms are applicable to any topologies.

[3]As an SFC in the real world consists of up to 13 VNFs [1], here we adopt the parameters from existing work only for the comparison purpose.

TABLE II
PERFORMANCE DIFFERENCES BETWEEN MCF AND SGA (%).

q_j	[0,0.04]	[0.04,0.08]	[0.08,0.12]	[0.12,0.16]	[0.16,0.20]
$r_i = 5$	3.5	8.0	5.2	9.2	12.7
$r_i = 10$	2.1	3.9	8.4	15.8	21.7

that MCF is an optimal algorithm that achieves maximum SFC availability. Second, we observe that for different algorithms, the resulting SFC availability decreases with the increase of q_j. This is evident as the more vulnerable the constituent VNFs, the less availability of the entire SFC. Third, it is interesting to notice that RSGA outperforms SGA in most cases by yielding slightly larger SFC availability.

Table II shows the performance differences between MCF and SGA from Fig. 4. We observe that it increases with the increase in the VNF failure probabilities of q_j. As the MCF algorithm is optimal and is more involved than a simple greedy algorithm like SGA, it is able to figure out the best backup assignment in a more stressful scenario of high VNF failures. Finally, it shows that this performance difference is larger in $r_i = 10$ than in $r_i = 5$. This shows our algorithm performs much better than existing work when there are more backup resources in the network. In particular, it shows that when $r_i = 10$, MCF outperforms SGA by up to 21.7% in terms of SFC availability.

VI. Conclusion and Future Work

We have proposed a fault-tolerant SFC framework named SAM that maximizes availability for SFCs considering both VNFs and backup server failures. In contrast, most of the existing research either just considered the failures of VNFs, or did not strive to maximize the availability for the entire SFC. We solved SAM by designing a minimum cost flow-based optimal and efficient algorithm and a suite of time-efficient heuristic algorithms. Via extensive research, we show our work outperforms the existing research considering the availability of both the VNFs and backup servers by up to 21.7%. This demonstrates the effectiveness of our algorithms in achieving high availability of SFC in cloud environments.

In future work, we will first extend SAM into the multiple SFCs case wherein different VNFs could need different processing capabilities. For example, a firewall VNF could cost twice as much memory as a NAT (network address translation) VNF [6]. Second, currently, we focused on stateless VNF updates, wherein the communication costs between VNFs and their backup servers are not considered. In the future, we will consider stateful VNF updates, wherein the stand-by VNF instances require constant state updates from active instances. As pointed out by [23], such state updates could consume considerable network bandwidth resources. How to maximize the SFC availability and minimize the bandwidth consumption of the VNF state updates under both VNF and backup server failures become a new challenging problem.

ACKNOWLEDGMENT

This work was supported NSF Grant CNS-1911191.

REFERENCES

[1] Service function chaining use cases in data centers (ietf). https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06section-3.3.1.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, 2008.

[4] F. Carpio, S. Dhahri, and A. Jukan. Vnf placement with replication for load balancing in nfv networks. In *IEEE ICC 2017*.

[5] R. Cziva and D. P. Pezaros. Container network functions: bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017.

[6] J. Fan, C. Guan, Y. Zhao, and C. Qiao. Availability-aware mapping of service function chains. In *IEEE INFOCOM 2017*.

[7] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao. A framework for provisioning availability of nfv in data center networks. *IEEE Journal on Selected Areas in Communications*, 36(10):2246–2259, 2018.

[8] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algorithms*, 22:1–29, 1997.

[9] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi. Traffic steering for service function chaining. *IEEE Communications Surveys Tutorials*, 21(1):487–507, 2019.

[10] F. He, T. Sato, and E. Oki. Optimization model for backup resource allocation in middleboxes with importance. *IEEE/ACM Transactions on Networking*, 27(4):1742–1755, 2019.

[11] H. Huang, S. Guo, J. Wu, and J. Li. Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing*, 7:1082–1094, 2019.

[12] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan. Reliability-aware virtualized network function services provisioning in mobile edge computing. *IEEE Transactions on Mobile Computing*, 19(11):2699–2713, 2020.

[13] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz. Optimizing virtual backup allocation for middleboxes. *IEEE/ACM Transactions on Networking*, 25(5):2759–2772, 2017.

[14] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz. Designing optimal middlebox recovery schemes with performance guarantees. *IEEE J. Sel. Areas Commun.*, 36(10):2373–2383, 2018.

[15] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue. Guaranteed-availability network function virtualization with network protection and vnf replication. In *IEEE GLOBECOM 2017*.

[16] S. Lal, T. Taleb, and A. Dutta. Nfv: Security threats and best practices. *IEEE Communications Magazine*, 55(8):211–217, 2017.

[17] J. Li, W. Liang, M. Huang, and X. Jia. Reliability-aware network service provisioning in mobile edge-cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, 31(7):1545–1558, 2020.

[18] R. J. Martins, C. B. Both, J. A. Wickboldt, and L. Z. Granville. Virtual network functions migration cost: from identification to prediction. *Computer Networks*, 181(9), 2020.

[19] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Sur. and Tut.*, 18(1), 2015.

[20] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz. A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks. *IEEE Transactions on Network and Service Management*, 14(3):554–568, 2017.

[21] X. Shang, Y. Huang, Z. Liu, and Y. Yang. Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme. *IEEE Transactions on Mobile Computing*, 2021.

[22] M. Wang, B. Cheng, S. Wang, and J. Chen. Availability- and traffic-aware placement of parallelized sfc in data center networks. *IEEE Transactions on Network and Service Management*, 18(1):182–194, 2021.

[23] B. Yang, Z. Xu, W. K. Chai, W. Liang, D. Tuncer, A. Galis, and G. Pavlou. Algorithms for fault-tolerant placement of stateful virtualized network functions. In *ICC 2018*.

[24] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu. Raba: Resource-aware backup allocation for a chain of virtual network functions. In *IEEE INFOCOM 2019*, pages 1918–1926, 2019.