On improving Service Chains Survivability Through Efficient Backup Provisioning

Saifeddine Aidi*, Mohamed Faten Zhani*, Yehia Elkhatib*‡
*École de Technologie Supérieure (ÉTS Montreal), Montreal, Quebec, Canada
‡MetaLab, School of Computing and Communications, Lancaster University, UK
E-mail: saifeddine.aidi.1@ens.etsmtl.ca, mfzhani@etsmtl.ca, {i.lastname}@lancaster.ac.uk

***Abstract*—With the growing adoption of Software Defined Networking (SDN) and Network Function Virtualization (NFV), large-scale NFV infrastructure deployments are gaining momentum. Such infrastructures are home to thousands of network Service Function Chains (SFCs), each composed of a chain of virtual network functions (VNFs) that are processing incoming traffic flows. Unfortunately, in such environments, the failure of a single node may break down several VNFs and thereby breaking many service chains at the same time.**

In this paper, we address this particular problem and investigate possible solutions to ensure the survivability of the affected service chains by provisioning backup VNFs that can take over in case of failure. Specifically, we propose a survivability management framework to efficiently manage SFCs and the backup VNFs. We formulate the SFC survivability problem as an integer linear program that determines the minimum number of required backups to protect all the SFCs in the system and identifies their optimal placement in the infrastructure. We also propose two heuristic algorithms to cope with the large-scale instances of the problem. Through extensive simulations of different deployment scenarios, we show that these algorithms provide near-optimal solutions with minimal computation time.

I. Introduction

The emergence of Network Function Virtualization (NFV) and Software-Defined Networking (SDN) technologies is currently transforming the way networks are designed and managed as they providing operators much more flexbility to dynamically provision network services and to dynamically configure the network. In particular, it is now possible to dynamically create chains of network services (Service Function Chains - SFCs) that can process incoming traffic and steer it across a chain of Virtual Network Functions (VNFs) like routers, IDS or NAT that are running on virtual machines.

In the last few years, a large body of work has been dedicated to address resource provisioning and management of such service function chains [1]–[5]. Most of existing studies assume the complete availability of the physical infrastructure which is not realistic as failures very are common in cloud network infrastructures [6]–[9]. Due to the dependency between virtual network functions in the chain, a single physical node failure in the network could easily bring down many VNFs and hence break several service chains and make several critical services unavailable. Such downtime, even for few seconds, not only hurt the reputation of service providers but also incur high revenue losses depending on the type of the offered service (e.g., 5600$ according to [10]).

Existing proposals to manage failures and mitigate them can be broadly categorized into reactive and proactive techniques [11]. In proactive techniques, backup VNFs are provisioned whenever an SFC is received and embedded. These backups remain idle but are activated only when a failure occurs to take over the service and replace the failed VNFs [12]–[15]. The second category of existing solutions are reactive techniques. These techniques do not pre-allocate backup resources and deal with failures after they occur [14], [16], [17]. Consequently, they need additional time to allocate resources and provision new VNF instances to take over the service. This definitely results in a longer service disruption, which is very costly for service providers [11]. This makes proactive techniques more appealing even though they may waste some resources for the backup VNFs.

To remediate to this problem and minimize wastage of resources, several research efforts advocate to use shared backup VNFs [13] where the same backup resource can be used to mitigate the failure of a set of VNFs assuming that they do not fail at the same time (i.e., only a single VNF from this set can fail at a time). In this context, this paper investigates possible solutions to ensure the survivability of service chains against single physical node failures by using shared backup resources. Unlike previous work addessing the same problem where backups are shared only between the VNFs of the same chain [12] [13] , our solution assumes that backup VNFs are shared among all the chains embedded in the infrastructure. This significanlty reduces the amount of resources used for the backup VNFs while still ensuring all SFCs are protected against single failures.

Our main goal is to ensure the survivability of all embedded SFCs against any single-node failure in the physical infrastrucutre. We reach this objective by procatively provisioning the minimal number backup VNFs to minimize resource wastage and by carefully placing them in the infrastrucutre. We also take into consideration the synchronization cost (in terms of bandwidth) and delay needed to keep the backup nodes up-to-date. We can summarize the main contributions of this paper as follows:

- We propose a resource management framework with a survivability module. This module allows to provision and manage backup VNFs and it could be

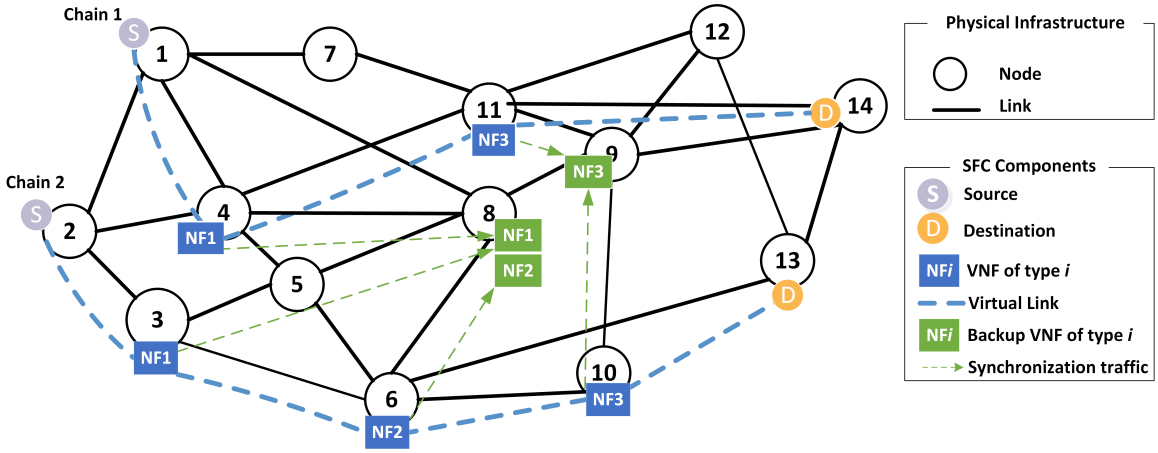

Figure 1: Example of embedded SFCs sharing backups

easily integrated into existing SFC resource management frameworks.

- We formulate the backup provisioning and placement problem as an Integer Linear Program (ILP) that finds the optimal number of backups for each type of VNF and optimally places them in the physical infrastructure.
- We devise two heuristic algorithms aiming at solving the problem for large-scale scenarios within a reasonable timescale.
- We evaluate the performance of the proposed heuristics and compare it to the optimal solution found with the proposed ILP solved by the CPLEX optimizer.

The remainder of this paper is organized as follows. Section II provides a detailed description of service chain survivability problem. We discuss relevant related work in Section III. In Section V, we mathematically formulate the addressed problem and then describe the proposed heuristic solutions. We present the experimental results in Section VI and follow up with some conclusions and future work in Section VII.

II. Problem Description

A Service Function Chain is made out from a set of different types of virtual network functions connected in a specific order to form a chain that steers the traffic from and to predefined source and destination [2]. A Virtual Network Function (VNF) is simply a virtual resource (i.e., virtual machine or container) that is running a specific network function (e.g., router, load balancer, NAT, IDS). To build the chain, the VNFs are connected through a set of virtual links having a sufficient amount of bandwidth to handle the traffic. Typically, service function chains are embedded into a physical infrastructure (referred to as NFV Infrastructure - NFVI [9]).

Fig. 1 shows an example of two service chains mapped onto a wide area NFV infrastructure. The figure shows for each chain how the VNFs are embedded from the source to the desination. For instance, chain 2 has traffic coming from physical node 2 towards physical node 13 and it is composed of three VNFs of type 1, 2 and 3 that are embedded in physical nodes 3, 6 and 10, respectively. The chain 1 has only two VNFs of type 1 and 3 that are embedded in physical nodes 4 and 11, respectively.

Once SFCs are embedded into the NFVI, the operator faces the challenging task of ensuring the high survivability of these SFCs. In other words, they need to survive potential network failures in order to minimize service interruptions. However, as mentioned earlier, physical node are very prone to failures and a single node failure may result in bringing down several VNFs and hence breaking multiple service chains. In this paper, we propose to ensure the survivability of the SFCs affected by a single failure by leveraging shared backups that could be used when the failure occurs. We also assume that a backup VNF should be of the same type of the set of VNFs it is backing up. In other words, a VNF of type i can only back up VNFs type i. This assumption is reasonable as in practice the backup is a virtual machine that should implement a specific software and hence a backup VNF has to contain exactly the same software stack as the original VNF.

As an example of how shared backups could be placed, we can see in Fig. 1 that physical node 9 hosts a backup of VNF type 3 (i.e., NF3) that is shared between the VNF type 3 of chain 1 and that of chain 2. If physical node 11 fails, and hence NF3 of chain 1 becomes out of service, the backup NF3 hosted in 9 takes over and replace the failed function. Similarly, it can take over the service of NF3 belonging to chain 2 (hosted in node 10) if it fails. The figure also shows other examples of shared VNF backups (e.g., NF1 and NF2 hosted in node 8). It is easy to check that the two service chains shown in this example are perfectly survivable to any single node failure.

Furthermore, backup VNFs are continously synchronized with the active VNFs to be ready to take over the service in case of failure (see green arrows in Fig. 1). For instance, the backup NF3 hosted in 9 has the state of NF3 hosted in physical node 11 and that of NF3 hosted in 10. Whenever a failure happens the last state of the failed function will

be used when the backup is activated. State synchronization can be done at different level. For example, at the level of the virtual machine running the function (e.g., memory synchonization [18]) or using customized synchonization scripts depending on the type of the network function (e.g., synchronizing rules in firewalls).

To make sure that the state synchronization is efficient, the latency between a VNF and its backup should not exceed a certain bound. Furthermore, synchonization of the VNF state may consume bandwidth that should be minimized. In our work, we ensure to minimize the synchonization delay and consumed bandwidth by limiting the number of hops between each VNF and its backup (for example, the number of hops is limited to 2 in Fig. 1).

The main challenge that we are addressing in this paper is how to find the minimal number of backup nodes and to determine their optimal placement in the physical infrastructure for each type of VNF taking into account the synchronization delay and cost in terms of bandwidth consumption.

III. Related Works

In this Section, we provide an overview of representative work on the survivability problem. We note that most of existing work focuses on virtual networks survivability and not service function chains. However, they still valid for our case as a service function chain can be seen as a particular case of a virtual network with a specific topology. In the following, we summarize existing techniques to ensure the survivability of SFCs and virtual networks. These techniques are either reactive or proactive [11].

Reactive techniques do not pre-allocate resources for backup but simply deal with a failure when it occurs. This leads to a long convergence time after the failure, resulting in a higher service downtime. On the other hand, proactive solutions anticipate failures and pre-allocate backup resources to ensure fast recovery of the service in case of failures.

Yu et al. [12] considered the case of a single-node failure and introduced two approaches to provision backup nodes. The first approach, called 1-redundant, redesigns the virtual network request into a survivable request by adding a single backup node. The second approach is called k-redundant where k is a constant that represents the number of backup nodes to be provisioned. The problem with these approaches is that a single redundant node may not be enough whereas k redundant nodes might be too much, and hence could incur a wastage of resources. To address this limitation, the solutions presented in this current work aim at finding the optimal number of the backup nodes when the proposed ILP is used or at least minimize it when the proposed heuristics are used.

In same direction, Ayoubi et al. [13] explored the space between 1 and k to find the optimal number of backup nodes to be incorporated into the requested virtual network. However, in this solution, the backup virtual nodes are provisioned for each request and hence they are not shared with other virtual networks. Our work is different in that it provisions backups that are shared among all virtual nodes belonging to all virtual infrastructures (i.e., SFCs) embedded in the physical infrastructure. As a result, our solutions further reduce the total number of the backups provisioned in the system.

Xiao et al. [16] proposed a topology-aware solution that ensures a rational resource allocation for the virtual network and a fail-over remapping based on a set of pre-computed detour-paths. Rahman et al. [14] proposed also a hybrid approach that benefits from a set of a possible backup detours for each link. These detours are proactively precomputed before the arrival of virtual network requests to allow fast re-routing in case of link failure.

Finally, Bo et al. [17] proposed a greedy algorithm that, in case of a link failure, searches for alternative resources to re-allocate the end-to-end path or re-embed the entire virtual network if resources are not sufficient. This may result in a long convergence time and a higher service downtime. In [20], Ayoubi et al. have demonstrated the NP-Hard nature of the survivability-aware embedding and proposed a polynomial time heuristic algorithm to restore failed services while maintaining the QoS requirements in terms of delays in case of single-node failures. Multiple failures were addressed in [21] where a heuristic was introduced in order to find a backup node. The algorithm is based on filtering techniques to parse the solution space and to speed up the search process for backups.

We summarize in Table I the aforementioned solutions. The table presents the type of technique adopted by each solution, indicates whether it is addressing a single or multiple failures, node or link failures and whether the backup are shared between the virtual nodes of all virtual networks or among the virtual nodes of the same virtual network. To summarize, the novelty of our work resides in the idea of sharing the backups between same type VNFs belonging to all service chains, which further reduces the amount of backup resources while still ensuring the survivability of the SFCs to any single failure.

In the following, we start by describing the proposed SFC management framework incorporating a survivability module beofre providing the details of the proposed solutions.

IV. Survivability Management Framework

In this section, we propose a management framework that incorporates the survivability management. Fig. 2 shows the main components of this framework. It is made out from the following modules:

- **Service Chain Provisioning Module:** this module allocates the resources for the service chains and instantiates the required virtual machines running the network functions. It also makes use of the SDN controller to provision the required amount of bandwidth and to set the required forwarding rules into the switches to streer the traffic across the VNFs composing each service chain. It is worth noting that the design of this module is out of the scope of this work. There is a large body of work addressing

Table I: Our solutions vs. existing solutions

Solutions	Type of solution		Single/Multiple failures		Node/Link failures		shared backup resources		shared backup among	
	Proactive	Reactive	Single	Multiple	Node	Link	Yes	No	all SFCs	single VN
Yu et al. [12]	x		x		x		x			x
Ayoubi et al. [13]	x		x		x		x			x
Guo et al. [19]	x		x		x			x		
Xiao et al. [16]	x			x	x			x		
Rahman et al. [14]	x		x			x		x		
Bo et al. [17]		x	x		x			x		
Ayoubi et al. [20]		x	x		x			x		
Ghaleb et al. [21]		x		x	x	x		x		
BS-Pull	x		x		x		x		x	

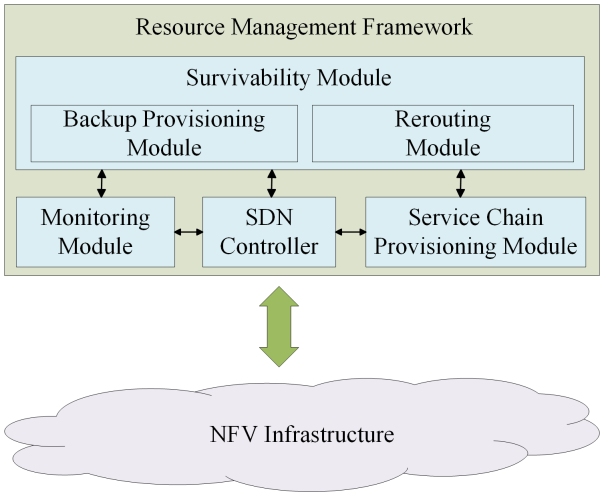

Figure 2: Resource Management Framework with Survivability Module

this module and any of the existing solution could be used (e.g., [1], [3], [5]).

• **Monitoring Module:** this module is in charge of continuously monitoring the infrastructure physical nodes and links and of feeding other modules at real time with the state of the resources. When a failure is detected, the monitoring module reports to the survivability module, which, in turn, reacts to mitigate the failure and ensure service continuity.

• **Rerouting Module:** in case of failure, the rerouting module redirects the traffic, that is originally destined to the failed VNFs, to the backup VNFs.

• **Backup Provisioning Module:** this module is responsible for finding the minimal number of backups needed to ensure the survivability of the embedded chains and to determine their locations. This modules also instantiate the backup VNFs and the synchronization links required to keep them up-to-date. In the following section, we describe in details of the proposed solutions to provision backup VNFs while achieving the sought-after objectives.

V. Backup Provisioning solutions

A. Integer Linear Program

In this section, we formulate the service chain survivability problem as an ILP aiming at minimizing the amount of resources allocated for the backup instances while ensuring a minimal synchronization cost and delay.

• **Infrastructure and chain Modeling:** we model the physical infrastructure as a graph denoted by $G = (N, L)$ where N is the set of physical nodes and L is the set of physical links connecting them. Each physical node $n \in N$ has a computing capacity c_n. The capacity c_n is expressed as the maximal number of VNFs that can be hosted by physical node $n \in N$. For sake of simplicity, we assume that each VNF is running on a single virtual machine with a standard size. We hence can see c_n as the maximal number of such virtual machines that could be provisioned in the physical node n. We also define d_{in} as the minimum number of hops separating the physical nodes i and n.

We model the service chain as a graph denoted by $S = (V, E)$ where V is the set of its composing VNFs and E is the set of virtual links connecting them. We assume there are different types of VNFs (e.g., Firewall, IDS, NAT). We denote by J the set of VNF types and we define m_{ij} as the number of VNFs of type $j \in J$ embedded in physical node i.

• **Decision Variables** we define two decision variables. We denote by x_{ij} the number of backup VNFs of type j embedded in physical node i. We also define $y_{ijn} \in \{0,1\}$ to indicate whether the backups provisioned for the VNFs of type j embedded in the physical node i are hosted in the physical node n.

• **Problem constraints:** in order to find a feasible solution, several constraints must be satisfied. For instance, to ensure that the primary and backup VNFs are not embedded in the same physical node, the following constraint must be satisfied:

$$y_{iji} = 0 \quad \forall i, j \in N \tag{1}$$

We also need to ensure that all VNFs of type j embedded in the physical node i necessarily have backups in another physical node:

$$\sum_{n \in N} y_{ijn} = 1 \quad \forall i, j \in N \tag{2}$$

Furthermore, if the backups of the VNFs embedded in the physical node i are hosted by the physical node n then the number of backups that will be provisioned in physical node n should be higher or equal than the number of VNFs. In other words, we have:

$$if \quad y_{ijn} = 1 \quad then \quad x_{nj} \geq m_{ij} \quad \forall i, j, n \in N \tag{3}$$

This if statement can be translated as the following constraint:

$$m_{ij} \leq x_{nj} + M(1 - y_{ijn}) \quad \forall i, j, n \in N \tag{4}$$

where M is a constant with a large value.

Furthermore, to ensure that the physical node hosting the backups have sufficient resources, the following capacity constraint must be satisfied for every physical node n:

$$\sum_{j \in N} m_{nj} + \sum_{j \in N} x_{nj} \leq c_n \quad \forall n \in N \tag{5}$$

where the first term represents the number of VNFs hosted in the physical node n and the second term represents the number of VNF backups hosted in the same physical node.

Finally, as we have to minimize the synchronization cost and delay between the VNFs and their backups, we limit the number of hops between each VNF and its backup to a limited number of hops denoted by d_{max}. Thus, we have:

$$if \quad y_{ijn} = 1 \quad then \quad d_{in} \leq d_{max} \quad \forall i, j, n \in N \tag{6}$$

The previous statement can be also written as the following constraint:

$$d_{in} \leq d_{max} + M(1 - y_{ijn}) \quad \forall i, j, n \in N \tag{7}$$

where M is a constant with a large value.

It is also worth noting that, for sake of simplicity, we assume that the number of hops between two physical nodes reflects the time delay between them. However, this may not be always true. In this case, our model can be easily updated to consider the propagation delay between the nodes by defining d_{in} as the delay of the shortest path between nodes i and n, and d_{max} as the maximum delay required between a VNF and its backup.

• **Objective function:** our ultimate goal is to minimize the amount of resources used by the backup VNFs while satisfying all the aforementioned constraints. This can be achieved by minimizing the total number of backups in all the physical nodes of the physical infrastructure. The objective function can be then written as:

$$\min \sum_{i \in N} \sum_{j \in J} x_{ij} \tag{8}$$

B. Heuristic Algorithms

In this section, we will present two heuristic solutions designed to solve the survivability problem. We denote the first algorithm as Backup Sharing "Pull" (BS-Pull) as we are looking at each physical node at see which VNFs it can back up (we refer to this as pulling). The second algorithm is called Backup Sharing "Push" (BS-Push) as we are doing the opposite which consists in taking each VNF and trying to find physical nodes that could host its backup (we refer to this as pushing) .

Both algorithms are carried out in two phases: finding the candidate physical nodes that satisfies the distance (for the sychronization delay) and the capacity constraints for a specific type of VNFs and then selecting among them to be the host of backups for a certain number of physical nodes. The difference between them resides in the way we are chosing the candidates and afterwards the backup hosts.

For the first algorithm BS-Pull, in the first phase, for each VNF type v, we start by associating each physical node p with a list of neighbors. We define neighbors as a the physical nodes that can be reached within d_{max} number of hops. Then, for each physical node p, the algorithm parses the list of neighbors and exclude the ones that cannot be served by physical node p. In other words, we remove the neighbors hosting a number of VNFs of type v superior to the remaining capacity in physical node p. In short, all physical nodes are candidates to host backups for their neighbors that can serve.

Algorithm 1 BS-Pull

Input:
$m_{i,v}$: Number of VNFs of type v in physical node i
Tot_n: Total number of VNFs in each physical node
Output: Physical nodes hosting backups

for all Type of VNF v **do**
  repeat
    for all $n \in N$ **do**
      for all $i \in N$ **do**
        if $d_{n,i} \leq d_{max}$ and $m_{i,v} + Tot_n \leq c_n$ **then**
          $neighbors_n \leftarrow \{i\}$
        end if
      end for
    end for
    for all $n \in N$ **do**
      $sum_n = sum\ of\ all\ m_{iv}\ /i\ in\ neighbors_n$
    end for
    Select the physical node n with highest sum_n to host the backups
  until all nodes are served or all resources are used
end for

For the second phase, we go through all the physical nodes and select the one with the highest number of VNFs of type v embedded in it neighbors. That is to say, we select the physical node that can serve the highest number of VNFs. This physical

node would represent the first host and the first cluster would be formed by associating it with its neighbors. We repeat the previous step without considering the physical nodes that are already affected to a cluster until all of them are served.

Algorithm 2 BS-Push

Input:
$m_{i,v}$: Number of VNFs of type v in physical node i
Tot_n: Total number of VNFs in each physical node
Output: Physical nodes hosting backups

1: **for all** Type of VNF v **do**
2: **for all** $i \in N$ **do**
3: **for all** $n \in N$ **do**
4: **if** $d_{n,i} \leq d_{max}$ and $m_{i,v} + Tot_n \leq c_n$ **then**
5: $candidates_i \leftarrow \{n\}$
6: **end if**
7: **end for**
8: **end for**
9: **repeat**
10: The physical node appearing in most of $candidates$ of each physical node i will be chosen as host for the backups
11: **until** all nodes are served or all resources are used
12: **end for**

As for the second algorithm, we are using another approach opposite to the first algorithm. In the first phase, for every VNF type v and for every physical node p, we parse all its neighbors to determine which ones can serve it. This means, we choose, from the neighbors, candidate physical nodes that have the capacity to host a number of backups equal to the number of VNFs embedded in the physical node p.

As for the second phase, in order to serve the highest number of VNFs while minimizing the number of backups, we take the physical node appearing in most of the candidates lists, in other words, the intersection of the physical nodes' candidates list that have common candidates. As for the physcial nodes without common candidates and still without backups, we repeat the previous step without considering those already served and affected to a cluster until all physical nodes are served.

VI. Simulation and Results

In this section, we compare the performance of the proposed algorithms with the optimal solution provided by CPLEX in terms of total number of backups and the execution time. To do so, we implemented the algorithms in C. We simulate the physical infrastructure and the service chain embedding. We have considered a network with 24 physical nodes with different capacities. The nodes are connected by 55 physical links that were randomly generated.

The objective of this set of experiments is to compare the number of backups and the number of VNFs left without backup provided by our algorithms with the one provided by CPLEX. We considered 8 different scenarios where the

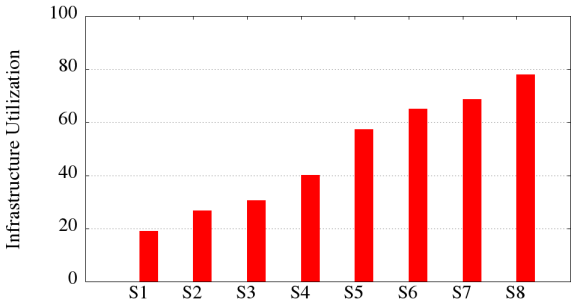

Figure 3: Infrastructure Utilization

utilization of the infrastructure as has been gradually increased as shown in Fig. 3.

A. Number of backups

Fig. 4 compares the total number of backups produced by our two algorithms with the optimal solution produced by CPLEX for each scenario.

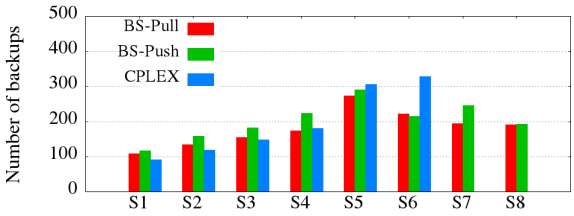

Figure 4: Number of Backups

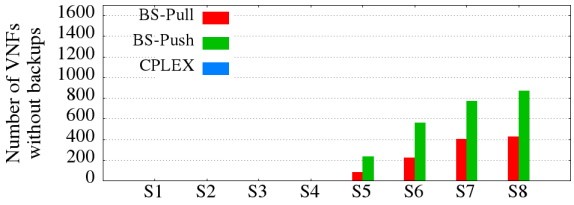

Figure 5: Number of VNFs without backups

The results shows that after a certain percentage of utilization CPLEX cannot provide a solution anymore, whereas, the two heuristics continue to provide results. However, as shown in Fig. 5, in some cases (e.g., starting from scenario 6), we notice that some of the VNFs are left without backups due to the lack of resources in the infrastructure. This is the same reason that prevents CPLEX from finding solution for scenarios 7 and 8.

As for the number of backups, the two heuristics provide a slightly higher number of backups compared to the optimal solution provided by CPLEX in low utilization scenarios, meaning that their solutions are not far from the optimal ones. In addition, we notice that the BS-Pull provides lower number of backups comapred to PS-Push.

B. Execution Time

Fig. 6 depicts the execution time of the two proposed algorithms compared to that of CPLEX for each of the 8 studied scenarios. The execution time for CPLEX goes from 2s to 7min (scenario 6) as the infrastructure utilization goes higher. This is because the number of variables in the ILP increase (e.g., the number of nodes and that of VNFs) and makes the problem harder to solve because of the large research space. It is also clear from the figure that the two algorithms execution time does not significantly change when the complexity of the system increases. We also note that , beyond the sixth scenario (i.e., scenarios 7 and 8), no optimal solution could be found.

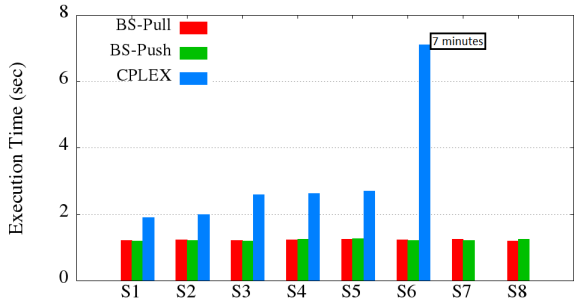

Figure 6: The execution Time

C. Synchronization Delay

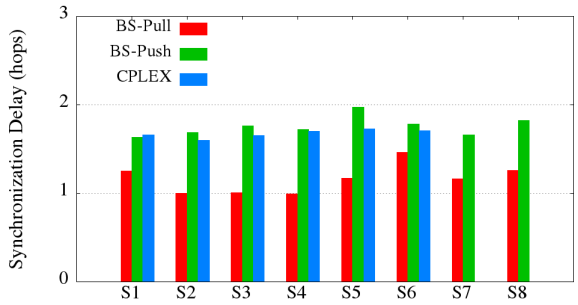

Figure 7: The synchronization Delay

Fig. 7 shows the average number of hops between a VNF and its backup for the different solutions and for the studied senarios. The number of hops provides some insight about the potential synchronization delay and its cost in terms of bandwidth. The results show that for the the two algorithms and CPLEX, the number of hops is below the maximal number of hops d_{max} specified as input to all solutions. However, we notice that BS-Pull provides lower delay compared to BS-Push and CPLEX.

VII. Conclusion

In this paper, we addressed one of the uprising challenges faced by the infrastructure providers: the survivability of the service chains against node failures. Despite recent researches on the problem, existing solutions present some major limitations whether it is a proactive solution where there is a waste of resources or reactive solutions where the convergence time can be very long.

We started by formulating the problem as an ILP and then proposed two heuristic algorithms to solve the problem for large-scale scenarios. Through simulations, we demonstrated that our algorithms performance is close to the optimal solution provided by CPLEX while they reduce the execution time considerably, although, for highly utilized infrastructures, some VNFs are left without backups.

References

[1] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 13, no. 3, pp. 518–532, 2016.

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

[3] W. Racheg, N. Ghrada, and M. F. Zhani, "Profit-driven resource provisioning in NFV-based environments," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[4] L. Qu, C. Assi, and K. Shaban, "Network function virtualization scheduling with transmission delay optimization," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2016, pp. 638–644.

[5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[6] J. Lee, H. Ko, D. Suh, S. Jang, and S. Pack, "Overload and failure management in service function chaining," in *IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–5.

[7] M. G. Rabbani, M. F. Zhani, and R. Boutaba, "On achieving high survivability in virtualized data centers," *IEICE Transactions on Communications*, vol. 97, no. 1, pp. 10–18, 2014.

[8] "Fail-slow at scale: When the cloud stops working," https://www.zdnet.com/article/how-clouds-fail-slow/, accessed: 2018-07-10.

[9] "ETSI GS NFV-REL 001 V1.1.1 (2015-01), Network Functions Virtualisation (NFV) Resiliency Requirements," https://goo.gl/PbQySQ, accessed: 2018-07-10.

[10] "Downtime, outages and failures - understanding their true costs," https://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html , accessed: 2018-07-13.

[11] M. F. Zhani and R. Boutaba, *Survivability and Fault Tolerance in the Cloud*. John Wiley & Sons, Inc, 2015, pp. 295–308. [Online]. Available: http://dx.doi.org/10.1002/9781119042655.ch12

[12] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *IEEE International Conference on Communications (ICC)*, 2011, pp. 1–6.

[13] S. Ayoubi, Y. Chen, and C. Assi, "Towards promoting backup-sharing in survivable virtual network design," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 3218–3231, 2016.

[14] M. R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[15] M. G. Rabbani, M. F. Zhani, and R. Boutaba, "On achieving high survivability in virtualized data centers," *IEICE Transactions on Communications*, vol. E97-B, no. 1, January 2014.

[16] A. Xiao, Y. Wang, L. Meng, X. Qiu, and W. Li, "Topology-aware virtual network embedding to survive multiple node failures," in *IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 1823–1828.

[17] L. Bo, T. Huang, X.-c. SUN, J.-y. CHEN, and Y.-j. LIU, "Dynamic recovery for survivable virtual network embedding," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 3, pp. 77–84, 2014.

[18] R. Boutaba, Q. Zhang, and M. F. Zhani, "Virtual machine migration: Benefits, challenges and approaches," in *Communication Infrastructures for Cloud Computing: Design and Applications*, H. T. Mouftah and B. Kantarci, Eds. USA: IGI-Global, 2013, pp. 383–408.

[19] B. Guo, C. Qiao, J. Wang, H. Yu, Y. Zuo, J. Li, Z. Chen, and Y. He, "Survivable virtual network design and embedding to survive a facility node failure," *IEEE Journal of Lightwave Technology*, vol. 32, no. 3, pp. 483–493, 2014.

[20] S. Ayoubi, C. Assi, L. Narayanan, and K. Shaban, "Optimal polynomial time algorithm for restoring multicast cloud services," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1543–1546, 2016.

[21] A. M. Ghaleb, T. Khalifa, S. Ayoubi, K. B. Shaban, and C. Assi, "Surviving multiple failures in multicast virtual networks with virtual machines migration," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 899–912, 2016.