On Optimizing Backup Sharing Through Efficient VNF Migration

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 emergence of software defined networking and network function virtualization technologies, network services are expected to be offered as service function chains made out from virtual network functions that are connected to steer and process the incoming traffic. In this context, achieving the survivability of these chains against failures is a key challenge to ensure high availability and continuity of the services. A promising solution proposed in the literature is to provision backups for the virtual network functions that could be shared among multiple service chains. These backups are used in case of a failure to take over the failed functions and ensure service continuity.**

In this paper, we propose two solutions to efficiently place and provision the shared backups in order to ensure the survivability of the service chains against single node failures. The originality of these solutions is that they leverage the migration of virtual network functions to minimize the resources consumed by the backups. Simulation results show that, compared to existing solutions, the proposed schemes leveraging migration are able to reduce by up to 20% the amount of resources allocated for the shared backups while ensuring the survivability of the service chains .

I. Introduction

The emergence of Network Function Virtualization (NFV) and Software-Defined Networking (SDN) is changing the way networks are designed and managed by offering more flexibility to customize and configure network services. These services are hence provisioned and implemented as Service Function Chains (SFCs) that process and steer traffic through Virtual Network Functions (VNFs) towards the destination.

In this context, one key challenge is to ensure the survivability of service function chains against failures. Indeed, the failure of even one single physical node hosting several VNFs belonging to different service chains would bring down these chains and affect their offered services. It is known that failures are common in cloud infrastructures. For instance, major cloud providers like Amazon, Microsoft and IBM have suffered in 2017 from several outages that could last to up to 4 hours, which affects their reputation and may translate into hundreds of thousands of dollars of revenue loss [1].

Existing literature contains a large array of proposals to manage failures and mitigate their impact in order to ensure service chains' survivability [2]. Previous work proposed to proactively allocate backups for VNFs so that they can take over when a failure occurs [3], [4]. They also advocate to use backups that are shared among multiple VNFs belonging to different service chains in order to avoid wasting resources. As such, one shared VNF backup could be used to mitigate the failure of multiple VNFs assuming that they do not fail at the same time, i.e., only a single VNF from these VNFs is assumed to fail at a time.

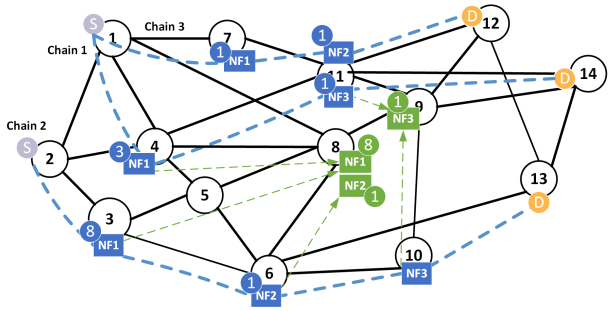
978-1-5386-9376-6/19/$31.00 ©2019 Crown

In this paper, we further investigate this solution. However, unlike previous work [2]–[4], we propose novel solutions that (1) decide on the number and the placement of the shared backups and (2) leverage VNF migration (i.e., the migration of the virtual machine or container hosting the network function) to relocate the VNFs in order to minimize the number of shared VNF backups.

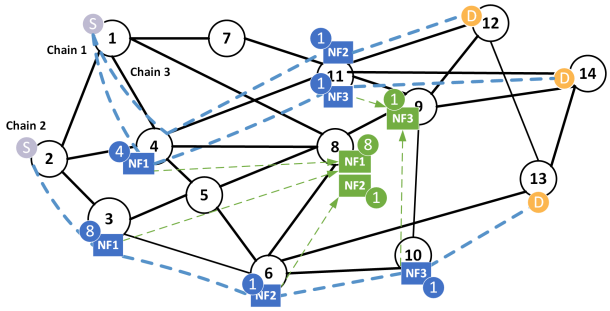
We can summarize the main contribution of this paper as follows:

- We devise a first backup provisioning scheme, called MABS-Pull, that starts by deciding of the number of shared backups and their placements and then uses migration to relocate the VNFs to ensure that all VNFs have backups.
- We devise a second backup provisioning scheme, called OBS, that leverages migration to relocate VNFs before allocating the shared backups and also after their allocation, which allows to further optimize the results.
- Through extensive simulations, we evaluate the performance of the two proposed schemes and compare them to an existing solution [3] that does not leverage VNF migration. Simulations show that OBS could reduce up to 20% the amount of resources allocated to shared backups while providing the same level of survivability achieved by the existing solution.

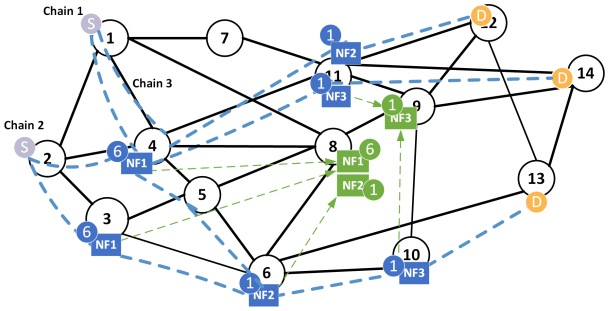
The remainder of this paper is organized as follows. Section II provides a detailed description of the service chain survivability problem and showcases how migration could be leveraged to reduce the amount of resources allocated for shared backups. We then discuss relevant related work in Section III. In Section IV, we present the proposed heuristic solutions. We describe the experimental results in Section VI. Finally, we present some conclusions and describe potential future work in Section VII.



(a) Original mapping of the service chain VNFs and their backups: the instance of NF1 hosted in 7 has no backup



(b) Mapping after the migration of the VNF NF1 from node 7 to 4: all instances of NF1 have now backups



(c) Mapping after the migration of 2 instances of NF1 from node 3 to 4: the number of needed backups for NF1 drops from 8 to 6

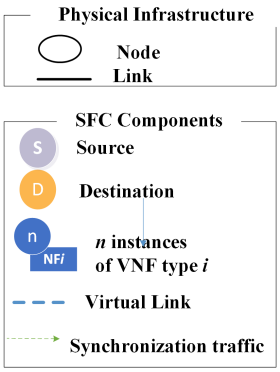


Figure 1: Example showing how VNF migration could allow (1) to ensure to have backups for all VNFs and (2) to reduce the number of backups

II. Problem Description

A service function chain is a set of different types of VNFs (e.g., router, load balancer, NAT, IDS). These VNFs are connected through a set of virtual links in a specific order to form a chain to steer the traffic from the source to the destination [5]. Service function chains are embedded into a physical infrastructure referred to as NFV Infrastructure (NFVI) [6].

Figure 1 shows an example of three service chains mapped onto a wide area NFV infrastructure and it shows also the backups that are provisioned to ensure the survivability against any single failure. The figure shows how the VNFs are placed from the source to the destination for each chain. For instance, chain 2 has traffic coming from physical node (2) towards physical node (13) and it is composed of three VNFs of type NF1, NF2 and NF3 that are embedded in physical nodes (3), (6) and (10), respectively. As an example of placing the shared backups, we can see in Figure 1 (a) that physical node (8) hosts 8 backup instances of VNF type NF1 that are shared between the 3 VNFs (type NF1) of chain 1 and the 8 VNFs (type NF1) of chain 2. If physical node (4) fails, the 3 VNFs (type NF1) of chain 1 become out of service, 3 backups type NF1 from those hosted in (8) take over and replace the failed functions. It is also easy to see that any single node failure could be mitigated using the provisioned backups. It is also worth noting that the VNF backups need to be continuously synchronized with the corresponding primary VNFs to save the last state of the failed function – see green arrows in Figure 1 (a).

Some VNF instances may be left without backups due to the lack of resources to provision backups. For instance, we can see in Figure 1 (a) that there is an instance of VNF type NF1 hosted in node (7) that has no backup. Figure 1 (b) shows that VNF migration could be leveraged to relocate the VNF that has no backups to other physical nodes where they can benefit from existing shared backups. The figure shows that the VNF type NF1 of chain 3 hosted in node (7) has been migrated to node (4), and thus, it is now backed up by one of the 8 backups already embedded in node (8).

Migration could be also leveraged to reduce the number of backups. This can be done if we distribute the VNFs fairly among the nodes that are close to the one hosting backups. Figure 1 (c) shows how relocating the VNFs could allow to

reduce the number of needed backups. In the figure, two VNFs type NF1 of chain 1 embedded in node (3) were migrated to node (4) and hence, with this new configuration, only 6 backups are needed rather than 8.

It worth noting that, in any of the three configurations (a), (b) and (c) illustrated in Figure 1, any single node failure could be mitigated using the provisioned backups.

We also note that migration has a cost (e.g., in terms of bandwidth and cpu consumption and service interruption as well [7]). To ensure a minimal migration cost, we assume in this work that a migration could be carried out only if the number of hops between the original location of a VNF and the new one does not exceed a maximal number of hops. For instance, the number of hops is limited to 3 in the example provided in Figure 1.

To conclude, as illustrated in the aforementioned example, the main challenge addressed in this paper is to leverage migration to ensure that all VNFs have backups and also to minimize the number of these backups by efficiently relocating the VNFs.

III. Related Work

In this section, we provide an overview of related work on the survivability problem. We also present some relevant work on VNF migration as our proposal relies on migration.

A. Backup provisioning

Reactive techniques do not pre-allocate backup resources but simply deal with a failure when it occurs which leads to a longer interruption time. As this work focuses on proactive techniques, we will focus mainly on proactive solutions.

Yu et al. [8] introduced two approaches to provision backup nodes to address a single-node failure. Both approaches redesign the virtual network request into a survivable network by adding backup nodes. The difference between the two approaches is the number of nodes added. The first approach, called 1-redundant, adds a single backup node whereas the second approach, called k-redundant, adds k backups nodes. The main limitation of this approach is the wastage of resources as the number of backups k is constant.

To address this limitation, Ayoubi et al. [4] aimed to find the optimal number of backup nodes to be added to the requested virtual network by exploring the space between 1 and k.

To further optimize the number of backups, we proposed in our previous work [3] two schemes, called BS-Pull and BS-Push, that provision backups that are shared among VNFs belonging to different service chains. These schemes are able to protect the chains again single node failures and significantly reduce the total number of backups provisioned in the system.

As all the aforementioned solutions do not leverage VNF migration to further optimize the placement and the number of backups requested to ensure the survivability of the service chains, this current work focuses on migration and shows how it can further reduce the number of the shared backups while achieving the survivability of the chains.

B. VNF Migration

Nobach et al. [9] propose SliM, a statelet-based framework for seamless VNF state migration that are based on a novel interface added to the VNF, sends statelets of the snapshot of the source VNF instance over the corresponding stream to the destination instance. This VNF state is responsible for replicating status of VNF instances of the same type in the service chain. Compared to the duplication-based model, that incur significant high additional costs according to them, SliM performs 3 times better in terms of link utilization. However, the authors did not take into consideration the distance between the VNF source and its destinations and the cost of the communication raised due to congested links.

Peuster and Karl [10] proposed E-State, a management framework that automatically handles the migration of the state of the VNF. The framework shares logically the VNF state by creating distributed state memory. Compared to centralized management system, E-State provide a better performance in terms of the number of replicated instances. One limitation of this work is that the authors do not take into consideration a service chain composed of multiple instances.

As the solutions proposed in this paper are based on our previous work [3], we provide in the following a brief overview of the the framework proposed in [3].

IV. Survivability-aware Management Framework

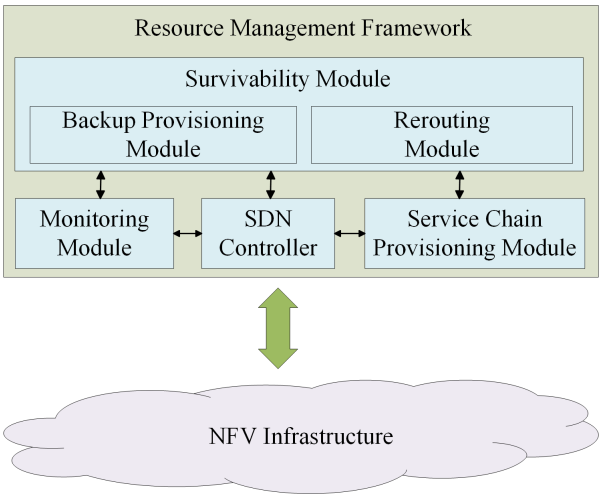


Figure 2: Architecture of the Proposed Resource Management Framework with the Survivability Module [3].

Figure 2 shows the survivability-aware service chain management framework proposed in [3]. In additional to traditional modules like the service chain provisioning module, the monitoring module and the SDN controller, we advocate to have a survivability module. This module is responsible for finding the minimal number of needed backups and for deciding where they should be provisioned (i.e., their locations) in order to protect the service chains against single node failures.

The module is implemented using the BS-Pull algorithm [3]. The BS-Pull algorithm identifies the number of shared backups for each types of VNF and decide of their placement in the physical infrastructures. It also minimize the synchronization cost and delay between the VNF and its backup by satisfying the two hop constraint, that is to ensure that the backup of a particular VNF is only two hops away from this VNF.

Unfortunately, with BS-Pull, some VNFs may remain without backups because of the lack of free resources to provision these backups or the inability to satisfy the two hop constraint. The solutions proposed in this paper aim at addressing this limitation by leveraging VNF migration in order to ensure that all VNFs have backups and that their placement satisfies the two hop constraint.

V. Migration-Aware Backup Provisioning

Before providing the details of the proposed schemes, we define a new metric called *the sharing ratio* as the total number of VNFs divided by the total number of their backups. If this ratio is high, it means that, on average, a high number of VNFs have a small number of shared backups. In other words, the higher is this ratio, the better is the provisioning scheme.

The objective of the migration in both proposed schemes is to maximize the sharing ratio by migrating the VNFs that are left with no backups. The first proposed scheme, called MABS-Pull (i.e., Migration After BS-Pull), migrates the VNFs after running the BS-Pull algorithm. The second proposed scheme is called OBS (i.e., Optimized Backup Sharing) and uses migration before and after applying the BS-Pull algorithm in order to optimize the number of backups to be allocated.

Both schemes are carried out in two phases. The first phase aims at finding the candidate physical nodes that satisfy the constraints of number of hops and the capacity. The second phase aims at migrating a certain number of VNFs to each of the candidate nodes. The difference between the two algorithms lies in the timing at which we execute the migration, as well as the objective of the migration itself. In the following, we provide the details of the two proposed schemes.

• **Algorithm MABS-Pull:** The scheme aims to distribute the VNFs that have no backups among the nodes whose VNFs have backups to further benefit from these existing backups. Assuming we consider VNF type $j \in J$ first, all nodes in the physical infrastructure are assumed to be able to host backups for type j VNFs. Our goal in the following steps is to select which node or nodes could really host the VNFs left without backups and how many VNFs per node.

We first define *the MigrateTo neighbors* of a physical node n (i.e., $MigrateTo(n)$) as the set of physical nodes that could be reached from n within at most d_{max} hops and such as the nodes in $MigrateTo(n)$ have some resources to host more VNFs. After executing BS-Pull for a certain type j, we compute the set of MigrateTo neighbors $MigrateTo(n)$ for each physical node $n \in N$ whose VNFs have no backups, and we also compute b_n, which is the number of VNFs of type j that should be migrated to $MigrateTo(n)$ in order to reach the number of backups already provisioned by BS-Pull. Finally, the VNFs are migrated and the whole process is repeated for all VNF types.

Algorithm 1 MABS-Pull

Inputs: Output of BS-Pull for each VNF type
N: set of physical nodes
u_n: total number of VNFs hosted in physical node n
m_{ij}: number of type j VNFs hosted in physical node i
$BNodes(j)$: set of physical nodes whose type j VNFs have already backups
$BHosts(j)$: set of physical nodes hosting type j backups
$x_{n_{host}}$: number of type j backups hosted in the node n_{host}
c_n: remaining capacity in each node $n \in N$
for $j \in J$ **do** ▷ *Parsing VNF types*
  for $n \in N$ **do**
    $MigrateTo(n) \leftarrow \emptyset$ ▷ *set of candidate nodes to host migrated VNFs from a physical node* n
    $b_n \leftarrow 0$ ▷ *number of type* j *VNFs to be migrated*
    ▷ *Finding candidate nodes to host VNFs migrated from* n
    for $i \in BNodes(j) \backslash (BHosts(j) \cup \{n\})$ **do**
      if $d_{ni} \leq d_{max}$ & $c_n \geq 0$ **then**
        $MigrateTo(n) \leftarrow MigrateTo(n) \cup \{i\}$
        $b_n \leftarrow x_{n_{host}} - m_{ij}$
      end if
    end for
  end for
  ▷ *Compute the number of migrated VNFs* s
  $s = \min(c_n, b_n)$
  ▷ *Migrate VNFs and update* $u_{n_{host}}$
  Migrate(s VNF type j, $MigrateTo(n)$, $u_{n_{host}}$)
end for

• **Algorithm OBS:** In this scheme, we are actually migrating the VNFs before allocating the backups. The goal is to minimize the number of backups provisioned. The migration is done after the first phase of the BS-Pull where we compute both the neighbors $SNeigh(n)$ and the number of backups to allocate.

The first phase of the migration is to determine the nodes having some resources left among $SNeigh(n)$ which will be refereed to as $MigrateTo(n_{max})$ where n_{max} is the node with the highest number of VNFs (i.e. the number of backups provisioned). The second phase consist in distributing some of the VNFs embedded in n_{max} among the nodes in $MigrateTo(n_{max})$ to get a balanced number of VNFs among all nodes. The backup resources are then recomputed, allocated and associated to all neighbors of the selected node.

Finally, we apply the MABS-Pull to maximize the coverage of the backups and the whole process is applied again for each

of the VNF types.

Algorithm 2 OBS

Inputs
N: set of physical nodes
u_n: total number of VNFs hosted in physical node n
m_{ij}: number of type j VNFs hosted in physical node i
c_n: remaining capacity in each node $n \in N$
for $j \in J$ **do** ▷ *Parsing VNF types*
  $BNodes(j) \leftarrow \emptyset$ ▷ *set of physical nodes whose type j VNFs have already backups*
  repeat
    Compute $SNeigh(n) \quad \forall n \in N$
      ▷ *Finding source neighbors of n*
    for $i \in SNeigh(n)$ **do**
      if $c_i \geq 0$ **then**
        $MigrateTo(n_{max}) \leftarrow MigrateTo(n_{max}) \cup \{i\}$
      end if
    end for
    ▷ *Compute the balanced number of VNFs s*
    $s = \frac{\sum m_{ij} / \ i \in MigrateTo(n_{max})}{|MigrateTo(n_{max})|}$
    ▷ *Migrate backups and update $u_{n_{host}}$*
    Migrate(s VNF type j, $MigrateTo(n)$, $u_{n_{host}}$)
    ▷ *Allocate backups and update $u_{n_{host}}$*
    Allocate (s backups, VNF type j, host n_{host})
    $BNodes(j) \leftarrow BNodes(j) \cup SNeigh(n_{host})$
  until $SNeigh(n) = \emptyset \quad \forall n \in N$
  MABS-Pull ▷ *execute the MABS-Pull algorithm for type j VNFs*
end for

VI. Simulation and Results

In this section, we compare the performance of the proposed backup provisioning schemes with the performance of the BS-Pull algorithm that does not use migration and that was proposed in [3]. The comparison is done in terms of total number of backups and the execution time and sharing ratio. To do so, the proposed algorithms were implemented in C. We simulated a physical network having 55 physical links and 24 physical nodes. These nodes have different hosting capacities ranging from 40 to 120 VNF instances.

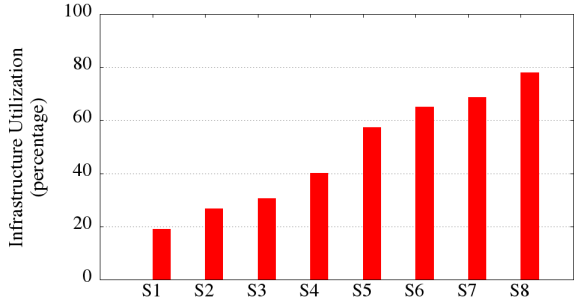


Figure 3: Infrastructure utilization for the studied scenarios.

We considered 8 different embedding scenarios provided by an existing VNF placement algorithm proposed by Racheg et al. [11] where the utilization of the infrastructure has been gradually increased. Figure 3 shows that we have low-utilization scenarios (i.e., S1–S4) where utilization is less than 50% and high-utilization scenarios (i.e., S5–S8) where utilization goes from 50% up to almost 80%.

A. Number of backups

Figure 4 compares the total number of backups found with the two proposed schemes compared to the original BS-Pull algorithm for each of the 8 scenarios. For low-utilization scenarios (i.e., S1 to S4), the number of backups provided by MABS-Pull is the same as the one provided by BS-Pull which is a normal as there is no VNFs left without backups after executing BS-Pull. However, OBS provides a slightly fewer number of backups compared to BS-Pull because the migration in OBS is carried out before the allocation of the backups aiming to optimize the number of backups for all tested scenarios.

For high-utilization scenarios (i.e., S5–S8), MABS-Pull provides a slightly higher number of backups whereas it is always less with OBS, which reduces by up to 20% the number of shared backups (e.g., scenario S6).

As to the number of VNFs left without backups (Figure 5), both MABS-Pull and OBS significantly reduce this number. We can see that there are VNFs left without backups for MABS-Pull for scenarios S5 and S6 while there are many for the last two scenarios S7 and S8. This is mainly due to the lack of free resource in the infrastructure to provision backups.

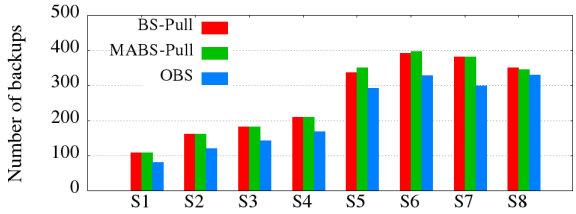


Figure 4: Total number of provisioned backups.

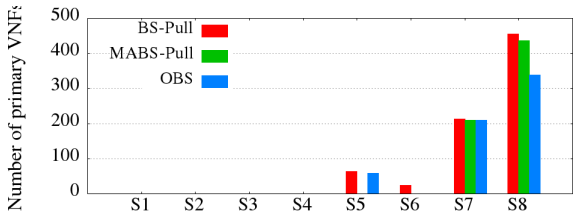


Figure 5: Number of VNFs without backup.

B. Execution Time

Figure 6 depicts the execution time of the two proposed migration schemes for each of the studied scenarios. The execution time of both schemes are slightly

different. OBS takes slightly more time than MABS-Pull as it is optimizing the number of backups by distributing the number of VNFs between the nodes before allocating the backups and migrating the left VNFs after the allocation.

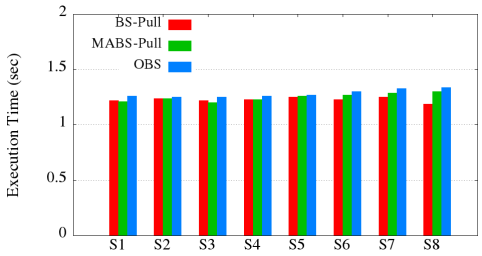


Figure 6: Execution time of the proposed solutions.

C. Infrastructure Utilization

Figure 7 shows the infrastructure utilization after the execution of the three schemes. Since MABS-Pull is providing the same number of backups as BS-Pull for low utilization scenarios (i.e., S1–S4), they both result in the same resource utilization. However, OBS leads to lower utilization as it optimizes the number of backups for all the studied scenarios. We can also see that, for high-utilization scenarios, the utilization remains high for the three algorithms, although OBS results in a slightly reduced utilization.

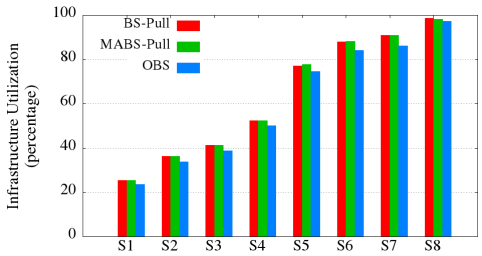


Figure 7: Infrastructure utilization after applying the different algorithms.

D. Sharing Ratio

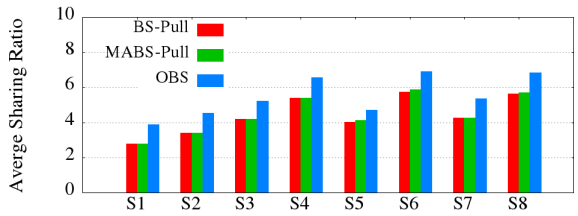


Figure 8: Average sharing ratio.

Figure 8 depicts the average sharing ratio found with the three studied schemes and for all the studied scenarios. The results show that starting from S5, we can notice a small difference between BS-Pull and MABS-Pull. However, for OBS, the sharing ratio is considerably higher than both solutions. These results demonstrate that using MABS-Pull and OBS allows to maximize the sharing ratio.

VII. Conclusion

Ensuring the survivability of service function chains is a challenging task for cloud providers. In this paper, we proposed two solutions to ensure the survivability of the service chains against single-node failures by proactively provisioning backups for the VNFs composing the chains. The originality of the proposed schemes is that they leverage VNF migration to reduce the number of the provisioned backups and to further optimize their placement. The conducted simulations showed that they are able to reduce by up to 20% the amount of resources allocated to VNF backups.

References

[1] "The 10 biggest cloud outages of 2017," https://www.crn.com/slide-shows/cloud/300089786/the-10-biggest-cloud-outages-of-2017-so-far.htm, accessed: 2018-07-10.

[2] 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

[3] S. Aidi, M. F. Zhani, and Y. Elkhatib, "On improving service chains survivability through efficient backup provisioning," in *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 108–115.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] M. Peuster and H. Karl, "E-state: Distributed state management in elastic network function deployments," in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*. IEEE, 2016, pp. 6–10.

[10] L. Nobach, I. Rimac, V. Hilt, and D. Hausheer, "Slim: Enabling efficient, seamless nfv state migration," in *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–2.

[11] 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.