CoShare: An Efficient Approach for Redundancy Allocation in NFV

Yordanos Tibebu Woldeyohannes, Besmir Tola, Yuming Jiang, *Senior Member, IEEE*
and K. K. Ramakrishnan *Fellow, IEEE,*

Abstract—An appealing feature of Network Function Virtualization (NFV) is that in an NFV-based network, a network function (NF) instance may be placed at any node. On the one hand this offers great flexibility in allocation of redundant instances, but on the other hand it makes the allocation a unique and difficult challenge. One particular concern is that there is inherent correlation among nodes due to the structure of the network, thus requiring special care in this allocation. To this aim, our novel approach, called *CoShare*, is proposed. Firstly, its design takes into consideration the effect of network structural dependency, which might result in the unavailability of nodes of a network after failure of a node. Secondly, to efficiently make use of resources, CoShare proposes the idea of *shared reservation*, where multiple flows may be allowed to share the same reserved backup capacity at an NF instance. Furthermore, CoShare factors in the heterogeneity in nodes, NF instances and availability requirements of flows in the design. The results from a number of experiments conducted using realistic network topologies show that the integration of structural dependency allows meeting availability requirements for more flows compared to a baseline approach. Specifically, CoShare is able to meet diverse availability requirements in a resource-efficient manner, requiring, e.g., up to 85% in some studied cases, less resource overbuild than the baseline approach that uses the idea of *dedicated reservation* commonly adopted for redundancy allocation in NFV.

Index Terms—NFV, redundancy allocation, shared reservation, network structural dependency, availability, reliability.

I. INTRODUCTION

NETWORK softwarization is transforming how networks are designed and operated to deliver specialized / innovative services and applications. Network Function Virtualization (NFV) has emerged as the key driver of network softwarization, promising among other aspects, network automation, flexible service provisioning, and cost reduction [1]. NFV is also considered as a key enabler for the new generation of communication networks such as 5G cellular networks [2], [3], which support diverse types of services, requiring different levels of availability. However, the successful adoption of NFV in production networks is associated with new challenges. One of them is to ensure these diverse availability requirements of services provided by an NFV-based network [4]–[6].

The "de-facto" technique for an NFV-enabled network to fulfill availability requirements of its services is through allocation of redundant/ backup network function (NF) resources

Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang are with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology, NTNU, Norway.

K. K. Ramakrishnan is with the Department of Computer Science and Engineering, University of California, Riverside, California, USA.

to compensate for the failures of primary NFs [5], since the mere provisioning of primary NF service chains is insufficient, especially for flows requiring carrier-grade services [5], [7], [8]. The diversity in the availability requirements of flows, implies that the levels of redundancy for them may differ. For example, while one backup chain might suffice to satisfy flows with lower availability requirements, more than one backup chain might be needed to meet higher availability demands.

One of the main factors which needs to be taken into account in redundancy allocation is efficient utilization of resources. In different virtualization technologies considered so far, redundancy is provided in the form of hot-standby replicas of NF instances. Typical solutions such as VMware Fault Tolerance [9] and the more recent NFV system-level framework [10] envision the instantiation of a *dedicated backup instance*, which runs *on a separate node*. However, such solutions can be resource inefficient, as each NF requires at least two instances (one primary and one backup).

Another crucial aspect that needs to be taken into account in redundancy allocation is the effect of the network structure or topology. This is due to that, the impact of one node's failure on the services provided by the network may significantly differ from that of another node's failure due to the inherent dependencies, called *network structural dependencies* [11], [12]. For example, failure of node n in Fig. 1 will cause three nodes to be unreachable and hence unavailability of all these nodes and their hosted NFs to other nodes in the network. As a result, for a service chain along with a backup chain, if it has an NF whose primary and backup instances were both hosted at these nodes, only one failure could cause both chains to become non-functional.

In addition, a unique characteristic of NFV is that *an NF instance may be hosted at any node in the network*. This appealing characteristic offers additional flexibility in redundancy allocation in NFV, i.e., the ability to decide *where to place the backup instances*, in addition to *how to assign backup instances* to form backup service chains for flows [13], [14]. While this allocation problem shares some similarity with the Virtual Network Embedding (VNE) problem [4], [14], as also highlighted in [14], specific facets such as allowing multiple flows to share a single virtual NF instance and requiring ordered concatenation of NFs to form a service chain make the problem different from VNE.

Previous works on NFV redundancy allocation consider also the problem of which nodes are assigned for backup instances, but without taking into account the freedom of placing backup instances at different nodes in the network

topology [8], [15]. More importantly, the problem of jointly addressing the placement and the assignment of service requests to backup instances has not been considered [16]–[20]. Approaches in [16], [21] provide node or NF instance level protection that is not tailored to the specific service availability requirements of individual flows. In works such as [17], [22]–[24], the fulfillment of service availability requirements at the flow or NF chain level is considered. However, the restrictive setups and assumptions made by them limit the scope and applicability of those approaches. For example, in [15], [24], backup instances of a service chain need to be hosted on the same node and in [8], [15], [25], dedicated capacity is reserved for every flow in the assigned backup instances. See the supplementary file for more detailed discussion about the related work.

In this paper, a novel redundancy allocation approach for NFV-based networks, called CoShare, is proposed. The contributions of CoShare are three-fold. Firstly, CoShare explores the flexibility offered by the unique characteristic of NFV, assigning backup NF instances to nodes meticulously to avoid the potential concurrent unavailability of the primary and backup service chains due to correlated failures caused by network structural dependencies, as exemplified above using Fig. 1. For this purpose, the information centrality measure called *node dependency index* [11], [26] is exploited to identify correlation among nodes (see Sec. III). The correlation information is made use of in both backup instance placement (§ IV-B) and assignment (§ V-A). To the best of our knowledge, CoShare is the first redundancy allocation approach for NFV-based networks, which explicitly considers the network structure-caused correlation among nodes in the design.

Secondly, exploiting the derived correlation information, CoShare proposes to improve resource utilization efficiency by exploiting *shared reservations*, where multiple service chains that are not susceptible to simultaneous service interruption (referred to as independent chains) are allowed to share the same reserved backup capacity of an NF instance. To the best of our knowledge, CoShare is the first to adopt this concept, i.e., *shared reservation*, in the NFV context. In the literature, the general idea of sharing backup resources has long been exploited for backup allocation to improve resource utilization in various types of networks, e.g., MPLS, IP, and optical mesh networks [27]–[33]. However, there is a fundamental difference. In traditional network settings, the locations of the backup resources are typically fixed in the network, while in an NFV-based network, owing to the flexibility offered by NFV, they cannot or need not be assumed a priori. In other words, the placement of backup NF instances can have a significant impact on the decision of how they can be shared, making the problem new and challenging. For NFV redundancy allocation, few works have considered exploiting sharing to improve resource utilization efficiency. Among the approaches described in the literature, either the sharing is at a coarser-grained level, e.g., NFs sharing a host machine [23], or it requires dedicated resources e.g. [24]. More discussion is provided in the Related Work section.

Thirdly, CoShare takes into account the heterogeneity in the availability requirement across flows in allocating back up NF
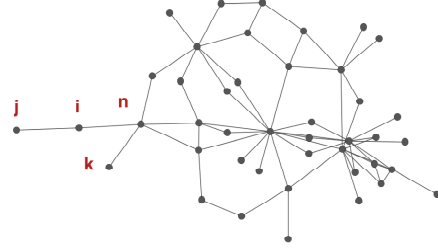

Fig. 1: Network-structurally correlated nodes with node i.

resources, together with the node heterogeneity in supporting NFs, where the heterogeneity in node and instance availability levels is also considered.

In brief, CoShare places backup NF instances and assigns them to form backup service chains to meet diverse availability requirements of flows by *jointly considering the network structural dependency-caused impact* (Sec. III, Sec. IV-B, Sec. V-A), *the heterogeneity in nodes, instances and availability requirements* (Sec. IV-B, Sec. V-A), and *the efficiency in resource utilization* (Sec. V-C). The main contributions of this paper are summarized as follows:

- A novel redundancy allocation approach for NFV, called CoShare, is proposed, where an information centrality measure is exploited to identify the inherent correlation among nodes due to the structure of the network (§ III), based on which backup NF instances are placed (§ IV-B) and assigned (§ V-A) to form the backup NF chains.
- The proposed heuristics of CoShare adopts a new concept of *NF shared reservation*, to achieve higher efficiency in resource utilization (§ V-C). Specifically, independent flows with disjoint primary service chains are allowed to share common reserved capacity at an NF instance for fault-tolerance.
- The placement and assignment of NF instances takes into account the heterogeneity of nodes, of instances, and required service availability levels, to decide where to place backup NF instances (§ IV-B) and how to assign NF instances to form backup chains (§ V-A).

II. THE SYSTEM MODEL AND PROBLEM DEFINITION

A. The Network Model

We consider the services provided by an NFV-based network, with the network represented as an undirected connected graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ the set of links. Each flow $f \in \mathcal{F}$, where $\mathcal{F}$ represents the set of traffic flows requesting service, has a source (s_f) and a destination (d_f), and has a traffic rate denoted as λ_f in packets per second (pps). The flow f is identified by the source-destination pair (s_f, d_f), where s_f and d_f are in $\mathcal{N}$. We call such source and destination nodes "end nodes" in this paper and assume that the end nodes are not involved in hosting NF instances. Each of the other nodes may have multiple CPU

cores to host NF instances. *An NF instance can be hosted on any node in $\mathcal{N}$, which is not a source or destination of a flow, and has enough available resources.* Keeping with typical deployment approaches [34], [35], we assume that a CPU core, if allocated, is dedicated to a single NF instance. An NF instance may be allocated one or more CPU cores.

The network service provided to flow f is represented by an NF chain $\overrightarrow{S}_f$, i.e., a series of network functions $(S_f^1, S_f^2 \ldots S_f^{g_f})$ that are performed in a specified order [4], where g_f denotes the service chain length. An NF instance v is assumed to require k_v number of cores and has μ_v processing capacity (i.e., the amount of traffic the instance can process per unit time). An NF instance may process multiple flows whose service chains include that corresponding NF. It is required that $\sum_{f \in \mathcal{F}^v} \lambda_f \leq \mu_v$, where $\mathcal{F}^v \subseteq \mathcal{F}$ denotes the set of flows processed by v[1].

It is assumed that for each flow, its service chain has already been allocated using an NFV resource allocation algorithm, e.g., ClusPR [35], where, however, the availability aspect of the service has not been taken into account. We term this allocated service chain as the *primary chain* of the flow.

B. The Availability Model

The network service provided to each flow f has an availability requirement A_{r_f}. For practical reasons, the failure impact of the flow's source and destination nodes is excluded. In addition, to simplify the representation and analysis, we focus on the impact of (hardware) node and (software) NF instance failures on the availability, and assume that nodes and instances fail independently. Reflecting the typical classification of availability requirements, as seen in the specification by the European Telecommunication Standard Institute (ETSI) [36], we consider three levels of service availability, referred to as classes. We assume that flows fall into: low, medium, and high availability class. As an example, in the evaluation, we assume the availability requirements of flows are in the range of $[99.9\%, 99.99\%)$, $[99.99\%, 99.999\%)$ and $[99.999\%, 99.9999\%)$ and use them to represent the three classes and also simply refer them as the 99.9%, 99.99% and 99.999% classes respectively.

We are interested in *meeting the availability requirements of flows through redundancy*, by allocating additional NF instances. More specifically, if the primary chain cannot provide the required A_{r_f}, *backup chain(s)* are created to improve the availability level of the service and meet the requirement. The primary chain of flow f, denoted with p_f, defines a mapping of the network service chain $\overrightarrow{S}_f$ with the nodes that host the service chain. Similarly, the backup chain b_f defines a mapping of the same chain to the nodes that will host the backup instances of flow f. Note that there can be more than one backup chains, hence mappings, if a single one is not sufficient to meet the flow's availability requirement. The service to a flow is considered available if either the primary or one of the backup service chains is available.

---

[1]If there are flows with extremely high traffic rates, NF instances with enough processing capacities should be included, or other mechanisms may be adopted, e.g., splitting the flow into sub-flows, to meet this requirement.

Let A_n denote the availability of node n, and A_v the availability of NF instance v. For the primary chain of flow f, we use A_{p_f} to denote its availability. For a backup chain b_f, its availability, denoted as A_{b_f}, is calculated as:

$$A_{b_f} = \prod_{g=1}^{g_f} A_{v_g} \prod_{n \in \mathcal{N}_{b_f}} A_n \qquad (1)$$

where $\mathcal{N}_{b_f} (\subset \mathcal{N})$ denotes the set of nodes that host NF instances of the backup chain b_f, and v_g the instance on b_f for the g-th NF of the service chain. If the backup chain(s) and the primary chain do not share a host node, the overall service availability A_f is given by a parallel combination:

$$A_f = 1 - (1 - A_{p_f}) \prod_{b_f \in B_f} (1 - A_{b_f}) \qquad (2)$$

where $B_f = \{b_f^1, \ldots, b_f^{h_f}\}$ denotes the set of backup chains and $h_f = |B_f|$ denotes the set cardinality.

C. The Redundancy Allocation Problem

For an NFV-based network, the redundancy allocation problem has three aspects to consider, which are,

- *(C1) Deciding the number of backup instance for each NF:* The needed number for each NF depends on the availability requirements of flows, the availability and capacity characteristics of both the nodes and the NF instances, as well as the topology of the network. When the availability requirements are stringent while nodes and/or NF instances have low availability values, more than one backup chain may have to be allocated [8], [37].
- *(C2) Placing the backup instances:* Similarly, the placement of backup instances is also influenced by the same factors. For example, the placement of a backup instance should not only comply to anti-affinity constraints [38] to avoid common failure modes, but it should also take into account the node's resource capabilities: placing a backup instance in a given node may happen only if the node has sufficient resources, e.g., CPU cores.
- *(C3) Assigning instances to form backup chains for flows:* This concerns the assignment of instances to each flow to form backup service chains to meet the flow's availability requirement. Similar considerations apply. For instance, a flow can be assigned backup instances only if those instances have sufficient resource capacity to accommodate the flow.

The three considerations discussed above are entangled, and together with the special NFV characteristic that any node may host instances of any NF, makes redundancy allocation in NFV both unique and challenging. To have a better overview of the problem and to achieve efficient resource utilization while providing redundancy allocation, we formulate it as an optimization problem as follows:

$$\text{Given:} \quad G(\mathcal{N}, \mathcal{L})$$
$$\text{Minimize}_{\forall (\beta, \alpha)} \quad N_v(\beta, \alpha), \qquad \forall v \in \mathcal{V} \qquad (3)$$
$$\text{Such that} \quad A_f(\beta, \alpha) \geq A_{r_f}, \quad \forall f$$

where β denotes the adopted NF backup instance placement strategy and α the assignment strategy: Each strategy specifies the values for the related decision variables, such as the number of instances for each NF and the node where an NF instance is placed. In addition, $\mathcal{V}$ is the set of NFs involved in providing the services, $N_v(\beta, \alpha)$ denotes the number of instances of NF v and $A_f(\beta, \alpha)$ is the achieved availability for flow f, under the strategies β and α.

For simplicity, only the topology and the availability conditions / constraints are included in (3). In our earlier work [39], a more complete version of the problem including the constraints can be found. In addition, an Integer Linear Program (ILP) model has been developed in [39] to solve the problem.

The optimization problem (3) is similar to the well-known Capacitated Location-Routing problem (C-LRP) that is NP-hard [13], [40], [41]. Specifically, our problem similarity to C-LRP consists in finding the optimal placement of NFs, among a set of available hosting nodes with different capacities, and finding the optimal paths for routing traffic along the NFs aiming at minimizing the total number of NFs and nodes to be utilized. A direct consequence of the problem complexity is that when the network is large, solving the problem optimally in limited time is difficult. For this reason, a heuristic approach, called CoShare, is proposed in this paper to address (C1) – (C3). The heuristic is introduced in detail in Sec. IV and Sec. V.

It is worth highlighting that *redundancy allocation for a backup chain should try to avoid sharing risk of failures with the primary service chain, i.e., the failure of any primary instance or its hosting node should have minimal impact on the backup service chain.* This is crucial and is also the basis for applying (1) and (2). However, even though the nodes and instances may be independent as individual systems, they are inherently correlated due to network structural dependence: the failure of one node may cause the unreachability of other nodes if these nodes have strong network structural dependency with the failed node (Cf. Sec. III). A novel idea of CoShare is to explicitly take into account the inherent correlation due to the network topology in the redundancy allocation problem (3).

III. IDENTIFYING CORRELATION AMONG NODES DUE TO NETWORK STRUCTURAL DEPENDENCE

A. Network Structural Dependency Measure

The inherent structural dependencies among nodes imply that the impact of one node's failure on the services provided by the network may significantly differ from the failure of another node. To reflect this difference, several measures have been proposed in the literature [11], [26]. For the NFV redundancy allocation problem, a key is to choose the proper nodes to place the backup instances. To this aim, the *node dependency index* [11], [26] is adopted.

The *node dependency index* $DI(i|n)$ measures the average level of dependency that node i has on node n in connecting to the other nodes of the network [11]. Specifically, $DI(i|n)$ is calculated from the path dependency index $DI(i \rightarrow j|n)$, which measures the dependency that the path between nodes i and j has on node n. $DI(i \rightarrow j|n)$ is defined as

$$DI(i \rightarrow j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } \mathcal{A}_{ij}^{-n} = 1 \\ 1 & \text{if } \mathcal{A}_{ij}^{-n} = 0, \end{cases} \quad (4)$$

where I_{ij} and I_{ij}^{-n} are the information measures between nodes i and j before and after the deactivation of node n [42]. Specifically, they are defined as:

$$I_{ij} = 1/d_{ij}; \qquad I_{ij}^{-n} = 1/d_{ij}^{-n}$$

with d_{ij} and d_{ij}^{-n} denoting, respectively, the length, in terms of hop count, of the shortest path between nodes i and j before and after the disabling of node n [42]. The binary variable $\mathcal{A}_{ij}^{-n}$ measures the reachability of node j from node i given that node n has failed: $\mathcal{A}_{ij}^{-n} = 1$ if node i can reach node j after the deactivation of node n and 0 otherwise.

The node dependency index is defined as [11] [2]:

$$DI(i|n) = \frac{1}{N-2} \sum_{j \in \mathcal{N}^{-n}/i \neq j} DI(i \rightarrow j|n). \quad (5)$$

where $N = |\mathcal{N}|$ and $\mathcal{N}^{-n} = \mathcal{N} \setminus \{n\}$ is defined as the set difference between $\mathcal{N}$ and $\{n\}$, i.e., the set of network nodes excluding n.

It can be verified: $0 \leq DI(i|n) \leq 1$. For the two extreme cases, $DI(i|n) = 0$ tells that i does not experience connectivity problem with removal of n, while $DI(i|n) = 1$ implies that i is unable to connect with any of the other nodes after n's failure.

Fundamentally, a higher $DI(i|n)$ value indicates higher dependence of i on n due to the network structure.

Remark: Having a longer path with more elements typically reduces the service availability. This motivates us to use (4) as the basis to measure how the reachability between two nodes depends on another node. Essentially, $DI(i \rightarrow j|n)$ quantifies the extent that this reachability from i to j is affected by the failure of node n, and hence the extent that $i \rightarrow j$ depends on n. Note that, except for (4), the other ideas of CoShare do not rely on the specific definition of $DI(i \rightarrow j|n)$, and hence could be readily applicable when other definitions for $DI(i \rightarrow j|n)$ are preferred.

B. Network-Structurally Correlated Nodes

As discussed above, even though individual nodes may fail independently, such a failure can affect the communication of other nodes in the network due to the inherent network structural dependence. From the definition of the node dependency index, i.e., (5), if node i has a higher-level dependency on node n, the failure of node n will result in greater difficulty for node i to reach the other nodes in the network.

We introduce $\mathcal{C}(i)$ as *the set of critical nodes of node i* which node i highly depends on. Node i is said to highly depend on a node $n \in \mathcal{C}(i)$, or in other words, n is critical to i, if $DI(i|n)$ is above a given threshold t_{DI}:

$$\mathcal{C}(i) = \{n \mid DI(i|n) > t_{DI}, n \in \mathcal{N}^{-i}\} \quad (6)$$

[2] In [11], $N-1$ is used in the denominator, but since there are only $N-2$ choices for $j \in \mathcal{N}^{-n}/i \neq j$, the more intuitive $N-2$ is adopted in (5). Note that, this change does not affect the resulting ranking of nodes.

Algorithm 1 Finding network-structurally correlated nodes

Input: $G(\mathcal{N}, \mathcal{L})$, t_{DI}
Output: $\hat{\mathcal{B}}_i$

1: Find $\mathcal{C}(i)$ using (6)
2: Insert $\mathcal{C}(i)$ to $\hat{\mathcal{B}}_i$
3: **for** $j \in \mathcal{N}^{-i}$ **do**
4:      **if** $i \in \mathcal{C}(j)$ **then**
5:          Insert j to $\hat{\mathcal{B}}_i$
6:      **if** $j \in \mathcal{C}(i)$ **then**
7:          **for** $k \in \mathcal{N}^{-j}$ **do**
8:              **if** $j \in \mathcal{C}(k)$ **then**
9:                  Insert k to $\hat{\mathcal{B}}_i$
10: return $\hat{\mathcal{B}}_i$

If $\mathcal{C}(i)$ is empty, it means that i is not highly dependent on the other nodes. For example, in a full mesh network, all nodes are structurally independent of each other as the failure of one node does not affect the connectivity among the others.

It is worth highlighting that the network structural dependence relation between two nodes, i and j, has two directions, i.e., $DI(i|j)$ – dependence of i on j and $DI(j|i)$ – that of j on i. As a consequence, to minimize the influence of network structure-caused correlation, such that node i can be used as a backup for node j and vice versa, we should avoid $j \in \mathcal{C}(i)$ or $i \in \mathcal{C}(j)$. We call this the *first-level dependency* among nodes. To elaborate, Fig. 1 shows an example, where nodes n and j have first-level dependency with node i. In particular, while $n \in \mathcal{C}(i)$ is critical to i, j is network-structurally highly dependent on i, i.e., $i \in \mathcal{C}(j)$.

In addition, as easily seen in Fig. 1, a critical node n of i may also be critical to another node k i.e., $n \in \mathcal{C}(i)$ and $n \in \mathcal{C}(k)$. In this case, both nodes i and k depend on the same node n. As a result, the failure of such a critical node, e.g. n, may result in the unavailability of those nodes that depend on it, e.g., i and k, hence presenting a structural correlation that we refer to as the *second-level dependency* among nodes. An implication of this is that: we should avoid using k as a backup for node i, even though k is not in $\mathcal{C}(i)$.

Based on the above analysis, Algorithm 1 presents the pseudo code of the algorithm for finding the set of nodes, denoted as $\hat{\mathcal{B}}_i$, which are network-structurally correlated with node i. $\hat{\mathcal{B}}_i$ is initially empty, the algorithm starts by finding the set of nodes based on the first-level dependency in both directions (Lines 1-2 and Lines 3-5 respectively). Then, nodes that have the second-level of dependency described above are added (Lines 6-9).

IV. PLACEMENT OF BACKUP NF INSTANCES

This section focuses on estimating the needed numbers of backup NF instances and deciding where to place them. Originally, we take network structural dependence into account.

A. Estimating Numbers of Backup NF Instances

The number of backup instances for each NF, needed to satisfy the service availability requirements of flows, is influenced by several factors, such as the availability and length of the primary chains and the availability and capacity of the NF instances, in addition to their availability requirements. In addition, flows with higher availability requirements might need more than one backup chain while lower service availability requirements may be fulfilled with only one backup chain [8], [18], [39]. Further due to the heterogeneity in node availability, instance availability, and service availability requirements of flows, finding the needed numbers of backup NF instances is not trivial. In CoShare, we use the following approach to estimate such numbers.

Specifically, we first estimate the number of backup chains needed to fulfill the availability requirement of flows in each class, based on which, the number of needed backup instances is then calculated. For the former, the estimation assumes that each NF instance is hosted at a different node and the backup chains and the primary chain are node disjoined. Then, the number of needed backup chains for a flow in class c, denoted as h_c, is estimated as:

$$h_c = \min_{x \in \mathcal{Z}^+} \left\{ x \mid 1 - (1 - \min_{f \in \mathcal{F}_c} A_{pf})(1 - \tilde{A}_c^b)^x \geq A_c^r \right\} \quad (7)$$

with

$$\tilde{A}_c^b = (\min_{n \in \mathcal{N}^b} A_n \min_{v \in \mathcal{V}} A_v)^g$$

where $\mathcal{F}_c$ represents the set of flows in class c and A_c^r the maximum availability requirement for flows in class c. $\tilde{A}_c^b$ may be interpreted as the availability of a backup chain for a class c flow, where $\mathcal{N}^b \subset \mathcal{N}$ denotes the set of nodes that have the capacity to host backup NF instances, $\mathcal{V}$ the set of NFs, A_n the availability of a node n, A_v the availability of an instance of NF v, g the maximum NF chain length of flows in the availability requirement class c. Since to satisfy the required availability more than one backup chain may be required, h_c is estimated from (7), taking into account the parallel effect of these backup chains.

Next, the number of backup instances of each NF v, denoted as z_v, which are needed to fulfill the service availability requirements of all related flows, is calculated as

$$z_v = \sum_c z_v(c) \quad (8)$$

where $z_v(c)$ denotes the number of backup instances of NF v which are needed for related flows in class c and is simply estimated from, assuming that all flows in the class require h_c number of backup chains:

$$z_v(c) = h_c \left\lceil \frac{\sum_{f \in \mathcal{F}_c / v \in \vec{S}_f} \lambda_f}{\mu_v} \right\rceil \quad (9)$$

Note that, the estimation (7) has adopted conservative assumptions, e.g. NFs of a chain are hosted at different nodes, to get (7). In comparison with the literature approaches in [17], [22]–[24], they may instead assume instances for the same NF chain to be hosted at the same backup node. In addition, with the estimation (8), it can be expected that the estimated numbers of backup instances are higher than the optimal numbers needed to fulfill the availability requirements of flows. However, we highlight that, the numbers from (8) are only "rough" estimates as the starting point. By setting the objective in assigning them to form backup chains to

be maximizing the utilization of the backup instances (cf, Sec. V), the actually used and hence needed numbers of backup instances can be significantly reduced (cf, Sec. VI-C1).

B. Placement of the Backup Instances

After the numbers of backup instances are estimated, CoShare places them on nodes. The heuristic is presented in Algorithm 2. The placement is made by performing *bin-packing* [43] of the NF instances on the nodes, where the *network structural correlation* among nodes, the *heterogeneity* in the availability level of nodes and NF instances, and in the availability requirements of flows, and the number of backup instances for each NF, are particularly taken into consideration.

On the node side for bin-packing, nodes are categorized and prioritized. Specifically, to avoid simultaneous unavailability of the primary and backup chains, it is intuitive that the NFs of a backup chain for a flow should avoid being hosted on those nodes that are "critically" correlated with the nodes hosting the primary NF chain of the flow. To this aim, based on the structural correlation information input, $\hat{\mathcal{B}}_n$, from Algorithm 1, nodes are categorized into two sets, $Q'_n(c)$ and $Q''_n(c)$, where the former represents the set of nodes that are not structure-critically correlated with the primary nodes of flows in class c, and the latter the rest. In the placement or bin-packing, as the intuition indicates, nodes in $Q''_n(c)$ are considered only after nodes in $Q'_n(c)$ have been exhausted (Lines 5 - 8).

In addition, nodes may have different availability levels, e.g., high-end nodes having 99.9% availability or higher while low-end nodes having 99% availability or lower. In [39], it is shown that for the low availability requirement class, it is more cost-efficient to use the low-end nodes. While for the medium and high availability requirement classes, using high-end nodes is preferable as this will lead to the use of fewer backup instances and nodes. Considering these, CoShare prioritizes nodes based on their availability levels. This prioritization is translated into the sorting of nodes (Lines 3 and 4), before the bin-packing is performed.

On the instance side for bin-packing, prioritization is also performed. Specifically, NF type that has greater number of backup instances to be placed is given higher priority to be placed. The underlying intuition is, the estimate (9) implies that more flows require this NF and its backup instances to achieve their availability requirements. Hence, giving it priority will more likely accommodate a higher number of flows [35]. This prioritization is reflected in sorting the NF instances based on their numbers (Line 1).

Finally, the placement is completed by bin-packing the backup instances onto the nodes, based on the prioritization introduced above. Specifically the top-prioritized node is checked for its capacity. If it has enough capacity, the NF instance with the highest priority is placed on it. If not, the next prioritized node is checked for possible placement of this instance (Lines 9-21). As a highlight, a node may have capacity to host multiple NF instances. In such a case, CoShare diversifies the types of instances placed on the node. The underlying intuition is that, the backup chain delay is minimized if all its NFs are hosted on the same node, and

---

Algorithm 2 CoShare's Placement Heuristic

Definitions:
$\mathcal{N}^c \leftarrow$ set of nodes hosting primary instances of class c flows
$Q'_n(c) \leftarrow$ priority queue of structurally uncorrelated candidate backup nodes
$Q''_n(c) \leftarrow$ priority queue of the other candidate backup nodes, i.e., $\mathcal{N}^{-Q'_n(c)}$
$Q_v(c) \leftarrow$ priority queue of all NF types v to be placed for each class c
$z_v(c) \leftarrow$ number of instances of NF type v to be placed for class c
Input: $G(\mathcal{N}, \mathcal{L})$, $\hat{\mathcal{B}}_n^s = \mathcal{N}^{-\hat{\mathcal{B}}_n}$ complement of set $\hat{\mathcal{B}}_n$ (from Algorithm 1)
Output: Placement of backup instances on nodes
Initialize:
$ActiveNode \leftarrow$ null
1: $Q_v(c) =$ sort $z_v(c)$ in descending order
2: **for** each class c **do**
3:     $Q'_n(c) = \bigcap_{n \in \mathcal{N}^c} \hat{\mathcal{B}}_n^s$ sorted based on node availability
4:     $Q''_n(c) = \mathcal{N}^{-Q'_n(c)}$ sorted based on node availability
5:     **if** $Q'_n(c)$ is not empty **then**
6:         $ActiveNode \leftarrow$ top of $Q'_n(c)$
7:     **else**
8:         $ActiveNode \leftarrow$ top of $Q''_n(c)$
9:     **while** $Q_v(c)$ not empty **do**
10:         $v \leftarrow$ NF type from top of $Q_v(c)$ with $z_v(c) > 0$
11:         **while** $Q'_n(c)$ and $Q''_n(c)$ not empty **do**
12:             **if** $ActiveNode$ has capacity **then**
13:                 Place v on the $ActiveNode$
14:                 $z_v(c) = z_v(c) - 1$
15:                 $v \leftarrow$ next NF type from top of $Q_v(c)$
16:             **else**
17:                 Remove $ActiveNode$ from $Q'_n(c)$ or $Q''_n(c)$
18:                 **if** $Q'_n(c)$ is not empty **then**
19:                     $ActiveNode \leftarrow$ top of $Q'_n(c)$
20:                 **else**
21:                   $ActiveNode \leftarrow$ top of $Q''_n(c)$

---

placing instances of different types on one node increases this chance. This is reflected by Lines 13-15 in Algorithm 2, where after placing a given NF type on a node, the next NF type from queue $Q_v(c)$ is chosen to be placed on the same node instead of another instance of the same type. This procedure is repeated until all the instances are placed or all the backup resources are utilized.

V. ASSIGNMENT OF NF INSTANCES TO FLOWS

The goal of redundancy allocation in NFV is to achieve the desired availability levels for flows. To this aim, having estimated the needed backup instances for each NF and decided where to place them in Sec. IV, CoShare assigns NF instances to flows to form backup chains for them so as to meet their availability requirements, which is the focus of this section. Some literature works also refer this problem as the flow routing problem or service chaining problem [41], [44].

A. Feasible NF Instance Set for Assignment

Note that every backup service chain of a flow must include all the NFs ordered in the same way as the primary chain. However, for every NF, it may have multiple instances. This subsection is devoted to identifying a set of such instances, called the "feasible set", which are considered for the assignment in CoShare. By *feasible set*, it is meant that with the backup chain formed by any combination of related instances in the set, the flow's availability requirement can be met.

It is obvious that any instance without enough capacity to accommodate the flow should not be included in the feasible set. In addition, as discussed in Sec. III, an NF instance whose

hosting node has critical structural correlation with a node of the primary chain (C.f. Algorithm 1) should not be included in the feasible set either. Let $\tilde{\mathcal{I}}_{S_f^g}$, $g = 1, \ldots, g_f$ denote the resultant instance set of each NF S_f^g of the flow.

Since each instance and the hosting node have implicit availability levels, this information can be made use of to find the feasible set. In particular, for one backup chain, the best availability range, which can be achieved, is between (MIN, MAX) which are calculated as

$$MIN = \min_{v_g \in \tilde{\mathcal{I}}_{S_f^g}, \forall g, \forall b} 1 - (1 - A_{p_f})(1 - A_{b_f}) \quad (10)$$

$$MAX = \max_{v_g \in \tilde{\mathcal{I}}_{S_f^g}, \forall g, \forall b} 1 - (1 - A_{p_f})(1 - A_{b_f}) \quad (11)$$

where A_{b_f} is found from (1). Note that (1) has two parts: $\prod_{g=1}^{g_f} A_{v_g}$ that is the availability resulted from the involved instances v_g and $\prod_{n=1}^{|\mathcal{N}_{b_f}|} A_n$ that is the availability resulted from the involved hosting nodes $\mathcal{N}_{b_f}$. When all instances v_g are hosted at different nodes, Eq. (1) can be re-written as

$$A_{b_f} = \prod_{g=1}^{g_f} A_{v_g} A_{n(v_g)}$$

where $n(v_g)$ denotes the node hosting the instance v_g. Intuitively, when the required availability A_{r_f} is smaller than MIN, this implies that all possible combinations out of $\tilde{\mathcal{I}}_{S_f^g}$ to form a backup NF chain for the flow are able to help meet the requirement and hence the set is already a feasible set. When $MIN < A_{r_f} < MAX$, it means the required availability can be achieved by some (but not all) combinations of instances in $\tilde{\mathcal{I}}_{S_f^g}$. In other words, this set is not a feasible set yet, and additional effort is needed as explained below.

CoShare uses Algorithm 3 to find the feasible set. To illustrate the idea, Fig. 2 shows a simple example. In the example, the considered flow needs a service composed of two NFs, firewall and load-balancer, and its availability requirement is 0.9999, but the primary service chain only has availability of 0.99. In the network, after excluding those instances without enough capacity left or whose nodes are network-structurally correlated with the primary chain nodes, there are still five candidate instances of each NF hosted at different nodes. After sorting, their availability levels, $A_{FW+} = A_{FW} \cdot A_n$ and $A_{LB+} = A_{LB} \cdot A_{n'}$, taking into account both node availability and instance availability, are shown in the figure.

Clearly, the required availability level 0.9999 is within this (MIN, MAX). An implication is that this requirement can be met with one backup chain. Another is that, some of the instances should not be included in the feasible set. To this aim, we drop the instance with the lowest availability, which is the load balancer instance with availability of 0.9, and then re-calculate MIN as shown in the middle part of Fig. 2 and compare it with the required availability. This process is repeated until MIN is equal to or higher than the required availability. All the remaining instances form the feasible set.

In the above discussion and example, the required availability level is less than MAX. If however this is not true, it means that one backup chain is not enough to fulfill the
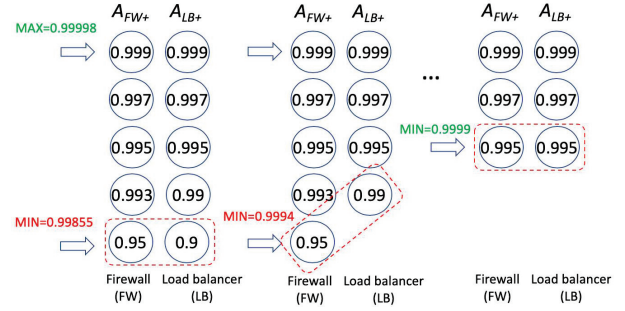

Fig. 2: Example of finding feasible instance set (Availability requirement: 0.9999; Availability of the primary chain: 0.99).

Algorithm 3 Finding feasible set of NF instances

Definitions:
$\mathcal{K}_v \leftarrow$ set of NF instances of type v with capacity for processing f
$A_{v+} \leftarrow A_v * A_b$, where b is the backup host node of NF $v \in \mathcal{V}$
Output: set of feasible NF instances $\{\mathcal{I}_{S_f^1}, \cdots, \mathcal{I}_{S_f^{g_f}}\}$

1: **if** $\mathcal{K}_v \neq \emptyset$ **then**
2:     Sort the instances in $\mathcal{K}_v$ in descending order of A_{v+}
3:     Find MIN from the instances that have min A_{v+} using Eq. (10)
4:     Find MAX from the instances that have max A_{v+} using Eq. (11)
5:     **if** $MIN \geq A_{r_f}$ & $|\mathcal{N}_{b_f}| = g_f$ **then**
6:         Insert the instances to the feasible set
7:     **else**
8:         **while** $(MIN < A_{r_f} \leq MAX) \,||\, (|\mathcal{N}_{b_f}| < g_f)$ **do**
9:             **if** $MIN \geq A_{r_f}$ **then**
10:                Insert the instances making min A_{v+} to the feasible set
11:             Replace the instance with the smallest availability
12:             Recalculate MIN
13:         Insert the instances to the feasible set
14: **else**
15:     Reject f

---

service availability requirement for the flow. In such cases, we update A_{p_f} with MAX in Algorithm 3 when applying (10) and (11), add the corresponding instances to the assignment and remove them from the candidate lists, and consider using an additional backup chain. This process is repeated until the required availability can be achieved, or there are not enough candidate instances left to form additional backup chains, implying the availability requirement is infeasible to achieve.

B. Feasible Backup Chains

Let $\{\mathcal{I}_{S_f^1}, \cdots, \mathcal{I}_{S_f^{g_f}}\}$ denote the feasible NF instance set, where $\mathcal{I}_{S_f^v}$, $v = 1, \ldots, g_f$, represents the set of instances of network function S_f^v in the feasible set. Then, the possible backup chains for the flow are easily obtained as:

$$(I_{S_f^1}, \cdots, I_{S_f^{g_f}}) \equiv \mathcal{R}(f) \qquad \forall I_{S_f^v} \in \mathcal{I}_{S_f^v}$$

where $v = 1, \ldots, g_f$ and $I_{S_f^v}$ denotes an instance of NF S_f^v. It is easily verified that, the total number of such possible backup chains is:

$$|\mathcal{I}_{S_f^1}| \cdots |\mathcal{I}_{S_f^{g_f}}|.$$

C. Assignment of NF Instances to Flows

In this subsection, we introduce the assignment strategy of CoShare. In brief, for each flow f, out of the feasible backup chains, CoShare assigns to the flow the chain that maximizes
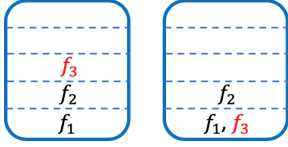
Fig. 3: Illustration of dedicated reservation (left) v.s. shared reservation (right): The instance has capacity of 5. Each flow requires capacity of 1. Flows f_1 and f_3 are independent.

the utilization of resources so as to minimize the total number of backup NF instances required.

1) Backup capacity reservation: Since each flow f has an arrival rate λ_f, every backup NF instance assigned to the flow needs to also reserve λ_f amount of capacity to the flow. As a consequence, it is intuitive to reserve the same amount of resource for every backup chain as for the primary chain, where dedicated capacity is reserved at every instance [8], [15], [17]. We call this approach *dedicated reservation*. Here, it is worth highlighting that, even in this approach, there is sharing at the instance level, i.e., the capacity of the instance can be shared by backup chains of multiple flows as long as the capacity constraint allows, i.e. $\sum_{f \in \mathcal{F}^v} \lambda_f \leq \mu_v$.

However, in practice, the probability that multiple independent failures occur at the same time is low and planning the redundancy considering this rare occasion is costly in terms of resource utilization [45]. Taking this into consideration, the idea of sharing reserved backup capacity among multiple flows has long been exploited in backup allocation to improve resource utilization in different types of networks [29], [31]–[33]. In CoShare, we propose to adopt the same idea to reduce the needed numbers of backup NF instances in redundancy allocation. We call this approach *shared reservation*.

Specifically, in CoShare, flows with disjoint primary service chains, referred to as *independent flows*, are allowed to share reserved capacity at a backup instance. Let $\mathcal{F}_v$ denote the set of independent flows whose service chains use backup instance v. Then, in CoShare with shared reservation, the backup NF instance v will only need to reserve a capacity of $\max_{f \in \mathcal{F}_v} \lambda_f$ for all these flows. In comparison, if dedicated reservation is used, for the same set of flows $\mathcal{F}_v$, the backup instance will need to reserve a capacity of $\sum_{f \in \mathcal{F}_v} \lambda_f$ for them. In Fig. 3, an example illustrating the difference in capacity allocation between shared reservation and dedicated reservation is presented.

2) The assignment heuristic: Note that for each flow, a set of feasible backup chains, denoted as $\mathcal{R}(f)$, which together with the primary service chain can meet the availability requirement of the flow has been identified in the previous subsection. Now, the challenge is how to apply the idea of shared reservation in this special setup, i.e. to choose / assign backup chain(s) to the flow to meet Objective (3). To this aim, CoShare utilizes a weight-based approach:

For each chain $r \in \mathcal{R}(f)$, it is given a weight $W^p(r, f)$ calculated as

$$W(r,f) = \sum_{g=1}^{g_f} w(v_{f,g}^r) \tag{12}$$

which is the summation of the weight of each instance constituting r, where $w(v_{f,g}^r)$ denotes the weight of the instance of the g-th NF in r for flow f, $v_{f,g}^r$ this instance, and g_f the service chain length for flow f.

In (12), the instance weight $w(v_{f,g}^r)$ is calculated using:

$$w(v_{f,g}^r) = \begin{cases} g_f, & f \perp\!\!\!\perp f_a \, \& \, f \perp\!\!\!\perp \mathcal{SR}_{v_{f,g}^r}(f_a), f_a \in \mathcal{D}_{v_{f,j}^r} \\ \dfrac{\sum_{f_a \in \mathcal{D}_{v_{f,g}^r}} \lambda_{f_a}}{\mu_{v_{f,g}^r}}, & \text{otherwise.} \end{cases} \tag{13}$$

where $\mathcal{D}_v$ denotes the set of flows that are reserving backup capacity at instance v, $\mathcal{SR}_v(f)$ the set of flows that are sharing the capacity of instance v with flow f, λ_f the rate of flow f, and μ_v the capacity of instance v. In addition, in (13), $\perp\!\!\!\perp$ is used to represent independence between flows.

The key idea of (13) is as follows. If flow f is independent with flow f_a, i.e. $f \perp\!\!\!\perp f_a$, as well as all flows that are sharing capacity of the instance $v_{f,g}^r$ with flow f_a, i.e., $f \perp\!\!\!\perp \mathcal{SR}_{v_{f,g}^r}(f_a)$, then it implies that flow f can *share reserved capacity* with flow f_a at the instance $v_{f,g}^r$. In this case, the instance will be given a weight equal to the service chain length of the flow, i.e. g_f. Otherwise, the instance is given a weight that is equal to the current utilization level of the instance.

In $\mathcal{R}(f)$, the backup chain that has the maximum chain weight i.e., $b_f = \max_{r \in \mathcal{R}(f)} W(r, f)$ is assigned for the flow f. This assignment strategy prioritizes the assignment of flows to instances that are utilized more, as implied by (13), so as to minimize the number of instances of each NF needed in Objective (3).

D. Efficiency and Scalability of CoShare

Theorem 1 ensures that the assignment heuristic of CoShare gives priority to feasible backup chains where shared reservation is possible, thus achieving higher resource efficiency.

Theorem 1. *Consider two backup chain choices r and r' ($\in \mathcal{R}(f)$). The other conditions are the same, but r has at least one instance $v_{f,g}^r$ on which the flow can share reserved capacity with other flows, while r' does not have. Then, r has a larger chain weight than r', i.e.,*

$$W(r, f) > W(r', f).$$

The complexity of CoShare is determined by the three involved parts, namely network structural analysis (Sec. III), placement (Sec. IV), and assignment (Sec. V). In brief, the complexity of CoShare can be written as a function of the number of flows that are to be assigned backup chains, the number of nodes / instances in the network, and the longest length of service chains, which is summarized as Theorem 2.

Theorem 2. *CoShare has a complexity of $O(FN^G)$, where F is the number of flows, G the longest length of service chains and N the number of nodes in the network.*

The proofs of Theorems 1 and 2 are included in the supplementary file.

VI. Results and Discussion

This section presents results showing the performance of the proposed redundancy allocation approach, CoShare. Recall that, a novel idea of CoShare is to exploit network structural correlation information in the design. Sec. VI-A is hence devoted to showing the impact of such correlation. The remaining Sec. VI-B – Sec. VI-D focus on introducing the performance of CoShare where comparisons are also included.

Specifically, a number of experiments are conducted on two realistic ISP network topologies. In the study, if not otherwise specified, it is assumed that each node hosts 8 CPU cores and has 16 GB memory, of which half of the capacity i.e., 4 CPU cores and 8 GB of memory, is used by the primary NFs and the rest by backup NFs. The primary NFs' placement as well as the assignment of primary service chains to flows is conducted by using ClusPR [35]. Every NF instance requires one CPU core and 2GB of memory, having a total NF processing capacity (μ_v) of 10Mpps.

Five types of NFs are considered (e.g., Firewall, DPI, NAT, IDS, and Proxy). The availability requirements of flows are set according to three levels, namely 99.9% (3'9s), 99.99% (4'9s) and 99.999% (5'9s). The NF processing capacity required by each flow f, i.e., λ_f, is set to 0.5 Mpps. The length of the service chain for each flow is assumed to vary in the range of 2 to 4. NFs and the service types in the chain of each flow are selected randomly out of the five NFs considered. The availability of the hosting nodes is assumed to be uniformly distributed between $0.99 - 0.999$, NF instances have an availability between $0.999 - 0.9999$, and the threshold algorithmic parameter t_{DI} is set to 0.5 if not otherwise specified. If a flow's availability requirement cannot be satisfied by the adopted redundancy allocation approach, it is rejected. All algorithms and simulations are run on a Dell workstation with a single Intel Xeon octa-core processor with 2.4 GHz base frequency and 64 GB of RAM. The results shown are average values with 95% confidence intervals over ten simulation runs for each availability level.

A. Impact of Network Structural Correlation

In this subsection, a simple experiment is carried out to showcase the effect of not considering the network structural correlation among nodes in the backup instance placement decision making. Two different network topologies are used, the GEANT network [46] and a simple Barabàsi-Albert (BA) scale-free network [47]. The GEANT network is a pan-European network connecting research and education institutions consisting of 44 nodes and 136 links. (To visualize this, a version with 44 nodes and 68 links is shown in Fig. 1.) The BA-network node linkage follows a power-law distribution and consists of 45 nodes and 45 links generated with an initial seed of $m_0 = 5$ nodes. Only node availability impact is considered and it is assumed that the availability of each node is 0.999 (3'9s). For each flow, the service function chain contains two NFs, and the required availability of the flow is 99.999%. The baseline algorithm from [8] is used to decide the number of backup instances needed. According to the baseline
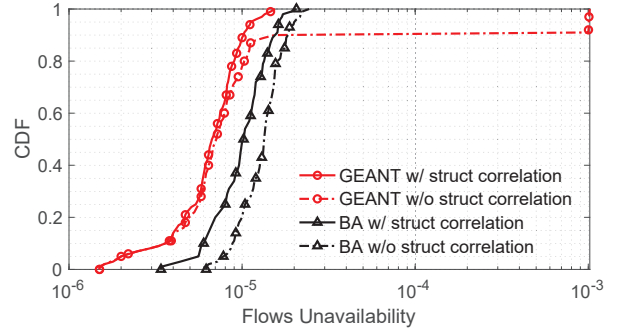

Fig. 4: Impact of Network Structural Correlation.

algorithm [8], theoretically, one backup for each of the NFs is enough to meet the 99.999% (5'9s) availability requirement.

Two backup instance placement strategies are considered. One does not consider network structural correlation, where the primary and backup NF host nodes of a chain are randomly chosen in the network. Another takes the structural correlation into consideration, where the backup host nodes are chosen from those without critical network structural correlation with the primary host nodes.

The availabilities of 100 flows are measured by conducting ten million simulation runs. In each simulation run, the state of each node, i.e., failed (0) or up (1), is randomly generated from Bernoulli distribution using the node's availability. The unavailability CDF of the 100 flows is shown in Fig. 4. As can be seen from the figure, the strategy taking network structural correlation into consideration performs significantly better for both networks. Specifically, when network structural correlation is not considered, about 10% of the flows only have availability of 3'9s or even lower for the GEANT network, in contrast to taking it into consideration where all flows have at least 4'9s availability. In addition, while only 80% of flows can reach the required 5'9s in the former, this percentage increases to 90% in the latter. Moreover, the effect of factoring in the structural correlation is more pronounced for the BA network scenario. All the flows achieve higher availability and 30% more flows are able to reach the target availability requirement of 5'9s compared to a placement strategy that does not consider structural correlation. The rationale behind this is due to the preferential attachment of BA networks, which tend to create highly connected nodes, also called hubs, whose failure can fragment the network into small isolated islands. From a robustness perspective, this property represents the Achilles' heel of scale-free networks [47]. As a result of such fragmentation, the reachability of nodes belonging to separate islands will be interrupted, hence an effective placement of redundancy needs to consider such effects. The ability of the structural correlation in capturing the reachability of nodes upon failure of others helps steer the backup placement more effectively.

There are two implications of this experimental study. One is that the number calculated by the baseline algorithm [8] may not be enough to meet the availability requirements of all flows when applied to a real network. Another is that, the inherent network structural correlation among nodes can

have significant impact on the availabilities of the services, and hence is a crucial factor that should be taken into consideration for redundancy allocation.

B. Comparison with Optimal Solution using ILP

In order to assess the optimality level of CoShare, its performance is compared with an integer linear program (ILP) approach called *AllAny*, which has been proposed to solve the optimization problem (3) in [39]. The ILP model assumes dedicated reserved capacity, i.e. dedicated reservation, at the backup instances. The GEANT network is also used for this analysis. Backup chains are allocated for 200 flows whose primary chains are served by using 23 NF instances, where every flow has a chain length of 2, and its availability requirement is randomly set to be 3'9s, 4'9s, or 5'9s.

For the comparison, we adopt the concept of *resource overbuild*, which is a key figure-of-merit in assessing redundancy capacity efficiency [31], i.e. the extra capacity needed to meet the service availability objective as a percentage of the capacity for the service under no redundancy. In this paper, it is defined to be the ratio of the total number of backup NF instances actually used to meet the availability requirement to the total number of primary NF instances.

Fig. 5 shows the placement of the backup instances, i.e., the number and type of backup NF instances placed together with their host nodes, obtained using the *AllAny* model and CoShare. For CoShare, both shared reservation and dedicated reservation are considered.

The number of backup instances for each NF which are created by CoShare with dedicated reservation is the same as that created by the *AllAny* model. In total, 23 backup instances are created by each of the two approaches and six backup nodes are used to host the instances, i.e., 100% resource overbuild. The per NF utilization level, which is the percentage of the NF capacity that is reserved by the flows, is shown in Fig. 6. Since CoShare intends to maximize the utilization of NF instances, most of the backup instances are 100% utilized in contrast to the NFs of the *AllAny* model. Moreover, with CoShare using shared reservation, only 13 backup NF instances are created and five backup host nodes are used, which results in only about 56% resource overbuild.

It is worth highlighting that in this example, the *AllAny* model is solved by using the mathematical programming solver, CPLEX. However, even for this simple example, it took more than 10 minutes to find the optimal solution, due to the NP-hardness of the optimization problem (3). In contrast, CoShare obtained the results in one second. This showcases the fact that CoShare is able to get near optimal results but in much less time, and when applying shared reservations, CoShare can achieve better resource efficiency.

C. Effect of Shared Reservation

We now consider a larger network: The Rocketfuel topology AS 1221 with 100 nodes and 294 links [48] is adopted. With the increased numbers of nodes and links, the computation of the optimal results has increased too much (due to NP-complexity) to be handled by the adopted workstation. For

this reason, CoShare with dedicated reservation, which has similar performance as the optimal model shown in the above example, will be used in the comparison.

1) Resource overbuild: Fig. 7a shows the number of backup NF instances that are needed to satisfy different levels of availability by using CoShare with shared and dedicated reservation, where all flows require the same availability level. In each of the availability levels, the NF instances are created for 700 flows having a service chain containing two NFs. In addition, the resource overbuild for both backup allocation strategies is illustrated in Fig. 7b. As can be observed, the higher the availability requirement level the more the number of backup NF instances required. This is because more than one backup chain might be required for fulfilling the high availability requirements. For example, more than 90% of the flows with five nines availability requirements are allocated two backup chains.

Observation: Recall that, in the placement phase, CoShare first "roughly" estimates the number of backup chains for each availability requirement class and accordingly the number of backup NF instances, i.e. h_c and $z(c)$ in Sec. IV-A, which are possibly needed. For the three availability levels [0.999, 0.9999, and 0.99999], the corresponding rough estimates are [1, 2, and 3] for h_c and [70, 140, and 210] for $z(c)$ respectively. As discussed in Sec. IV-A, the actually used and hence needed numbers of backup instances under CoShare can be expected to be much lower. This is confirmed by Fig. 7a that shows the real total numbers of used backup instances under CoShare with dedicated reservation and shared reservation. Specifically under CoShare with dedicated reservation, they are [68, 85, 130] for the three availability levels, which are reductions of 2.8%, 39%, and 38% from the rough estimates respectively. Under CoShare with shared reservation, there are further reductions of 44%, 64%, and 69%.

Finding: Fig. 7b further compares dedicated reservation and shared reservation using resource overbuild. As shown by the figure, to fulfill 0.99999 availability, the former requires 178% resource overbuild, in contrast to the much reduced 93% by the latter. Similar reduction is found also for the other two availability levels. Overall, shared reservation enables more efficient utilization of resources with significant decrease in the required number of backup NF instances.

2) Flow acceptance ratio: In the above experiments, no flow is rejected, i.e. all flows' availability requirements can be met with CoShare, with or without shared reservation. In the following experiment, we consider 650 flows each with a service chain consisting of four NFs. Similar to Fig. 7a, Fig. 8a compares the number of backup NF instances under dedicated and shared reservation. It also shows that less instances are needed with shared reservation.

Additionally, Fig. 8b compares the flow acceptance ratio. As shown by the figure, all flows can be admitted with both shared and dedicated reservation when the availability requirements are under the two lower levels. However, when 5'9s availability level is required, only about 60% of the flows can be admitted with dedicated reservation, in contrast to 100% with shared reservation. This is because *CoShare with dedicated reservation* requires a higher number of backup NF
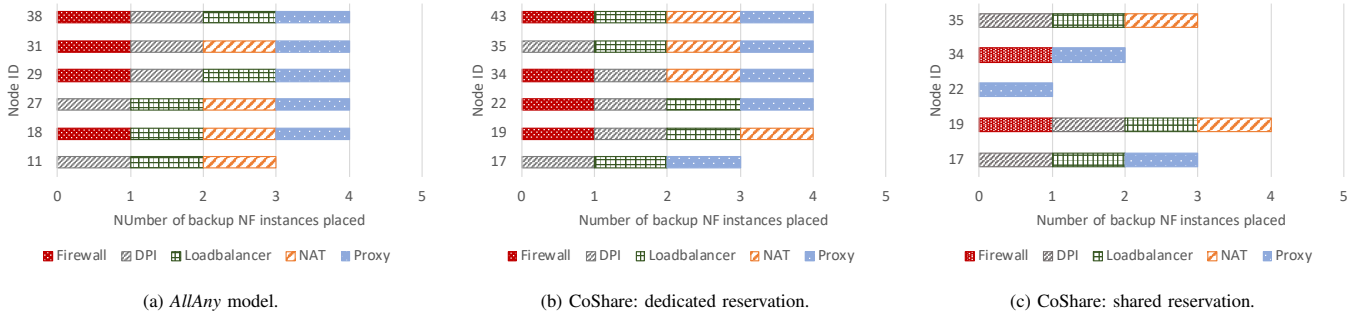
(a) *AllAny* model.

(b) CoShare: dedicated reservation.

(c) CoShare: shared reservation.

Fig. 5: Placement of backup NF instances by the *AllAny* model [39], and CoShare using dedicated and shared reservations.

(a) *AllAny* model.
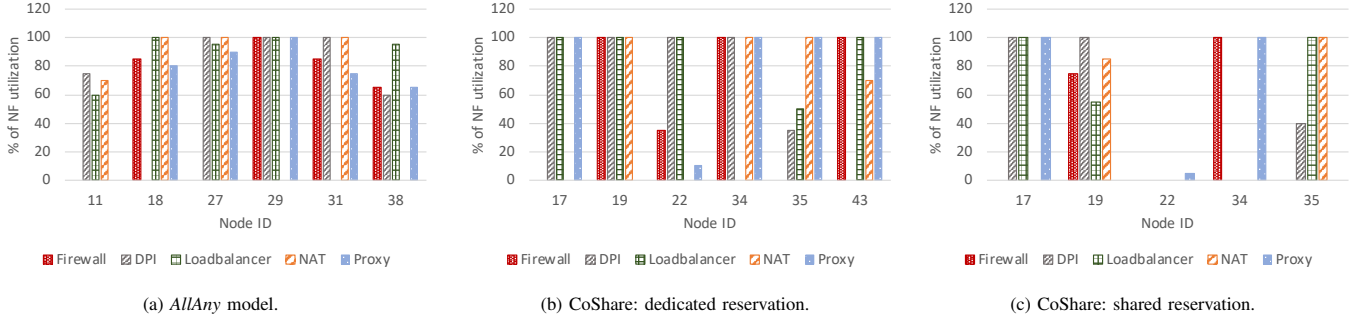
(b) CoShare: dedicated reservation.

(c) CoShare: shared reservation.

Fig. 6: Per NF utilization of the placed backup NF instances.

(a) Number of backup NF instances.

(b) Resource overbuild.

Fig. 7: Comparison: Resource overbuild.

(a) Number of backup NF instances.

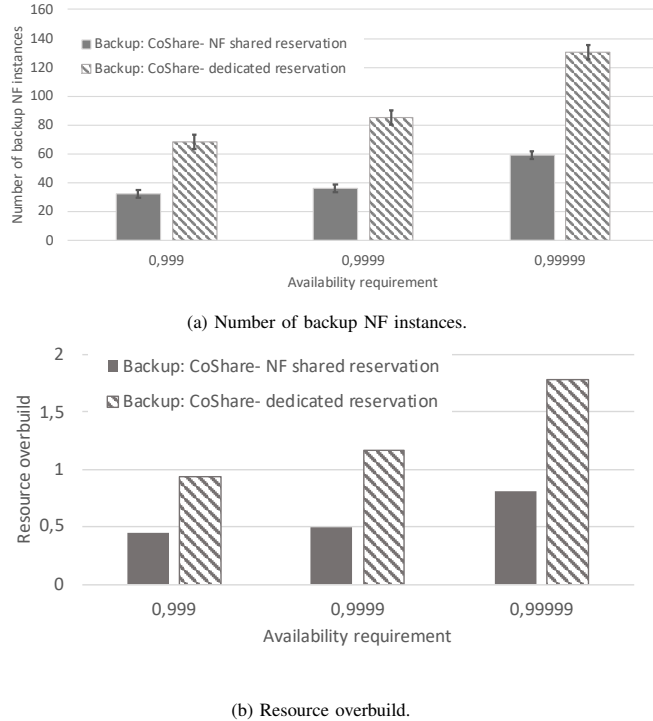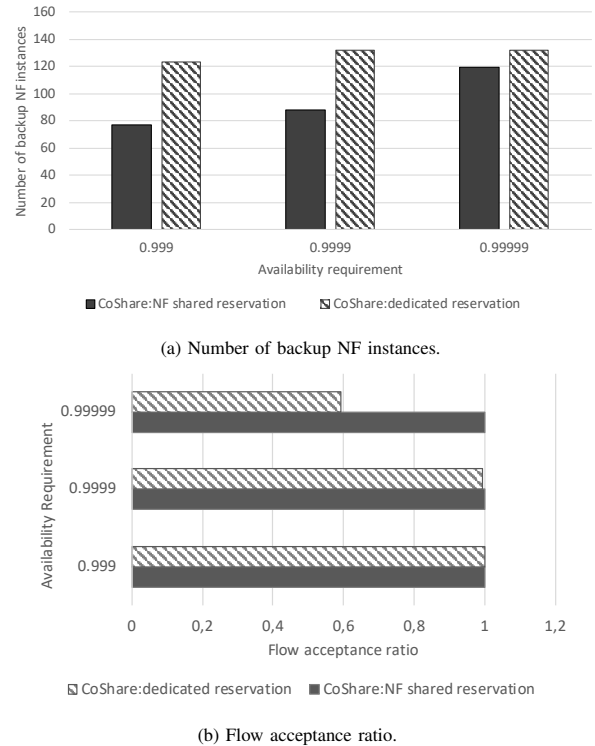(b) Flow acceptance ratio.

Fig. 8: Comparison: Flow acceptance ratio.

instances than what can be provided by the network. If that number would have been possible, Fig. 8a would have shown an even higher reduction by using shared reservation. This again implies that NF shared reservation enables more resource efficiency which in turn maximizes the number of flows that can be admitted to the network.

D. Effect of the Threshold

CoShare has one algorithmic parameter, the threshold $t_{DI} \in (0, 1)$, which is used in (6) to help identify the set of critical nodes due to network structural correlation. As can be expected from (6), a higher t_{DI} leads to a smaller set. To have a

better overview about the effect of the threshold, experiments have also been conducted. Figure 9 shows the number of backup NF instances instantiated for fulfilling the availability requirement of 0.99999 under different threshold values for the case of Rocketfuel topology with 700 flows. For the threshold value between 0.2 and 0.9, the same number is found. *With a closer look, it has been found that the same set of structurally correlated nodes are resulted from (6) with a threshold value in this range, which constitutes the underlying reason for the effect shown by Figure 9.* However, when $t_{DI} \in (0,1)$ is too small, e.g. 0.1, it is observed that all the network nodes are included in the set. These indicate that the performance of CoShare is generally robust to the threshold except when a too small value is given.
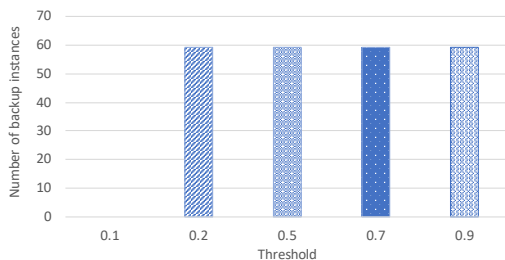

Fig. 9: Effect of t_{DI}.

E. Effect of Assignment Order

Note that when CoShare is applied, flows are checked one-by-one in the step of assigning instances to form back chains for them (C.f. Sec. V). When there are multiple flows, these flows may be applied with CoShare at arbitrary order. In the rest, the effects of two specific yet intuitive orders are investigated.

In this investigation, the Rocketfuel topology is adopted. 15 sets of randomly generated 700 flows are considered. Each set is used for one simulation run. The availability requirement of each flow is randomly chosen among 3'9s, 4'9s and 5'9s. CoShare with shared reservation is focused.

1) Based on availability requirement: First, we consider the effect of ordering the assignment based on the required availability levels. In this study, flows have the same service chain length of two. The flows are assigned with needed backup instance in the order of their availability requirements, from high (5'9s) to low (3'9s) or from low to high.

Fig. 10a compares the total needed number of backup NF instances in fulfilling the availability requirements of flows in the 5'9s class under each simulation run using a different set of flows. As can be seen from the figure, there is no clear evidence about which order is better. Similar observation has also been found for the other two classes. This implies that prioritizing assignment based on availability requirements has minimal effect on the performance of CoShare.

2) Based on service chain length: Next, the impact of ordering based on the flow's service chain length is assessed: long to short or short to long. The experiment setup is similar to above, but the chain length of a flow is randomly chosen between 2 and 4.

Figs. 10b and 10c compare the number of needed backup NF instances for the 3'9s and 5'9s classes respectively. As can be observed from these figures, prioritizing flows that have longer service chain length results in fewer backup NF instances needed. One reason is that, when a longer chain flow is prioritized because of the involvement of more instances, the chance for a shorter flow that is assigned later finding and sharing capacity on some of these instances is higher. For example, in the best case, no new backup instance is needed. Nevertheless, the difference is minimal, about 10% or less.

F. Discussion

The evaluation highlights the significant improvement that CoShare brings in terms of resource utilization efficiency and flow acceptance ratio. However, generalizing the results requires performance evaluation in more diversified NFV setups. We briefly present some aspects that could be further explored as future work.

The placement decisions are subject to the underlying network topology structure and its intrinsic graph characteristics (i.e., node degree, betweeness etc.). Aggregated results on a variety of network topologies, e.g., those present in SNDlib [49], may provide more detailed insights about the effects of structural correlation on the placement and performance. In addition, it is common that NFV-enabled network operators provide services through well-defined service chains, typically included in their NFV service catalogs, rather than through a random composition of NF instances. An alternative evaluation would be to consider a set of specific service chains and characterize their availability demands and probability of occurrence, i.e., probability of being requested. Moreover, we made some simplifying assumptions compared to a more truthful representation of network nodes' and links' capacities. In realistic networks, some of the nodes could be central offices with rather limited capabilities, whereas some other nodes could be data centers with significantly more computing resources. Besides, we have implicitly assumed that all links have the needed capacity. However, failure events may trigger traffic re-routing, increasing the utilization levels of certain links. A related aspect concerns cases where multiple distributed copies of an NF on the service chain may be needed to work together, to deliver the service, e.g., a load-balancer. To address these needs, CoShare may be extended to include link utilization constraints and to consider more generalized ways of defining service chains. Clearly, there will be an increase in the problem complexity, as well as a modification of the assignment heuristics to account for the link utilization constraints and to support multiple copies of one NF on a chain. This could be another direction for future extension of CoShare.

VII. CONCLUSION

In this paper, a novel scheme, called CoShare, is proposed for redundancy allocation in NFV. An original and crucial idea of CoShare is to explicitly take into account the inherent network structure-caused correlation / dependence among nodes in both redundancy placement and assignment. As a result,

(a) Chain length=2.

(b) Low Availability class: 0.999.
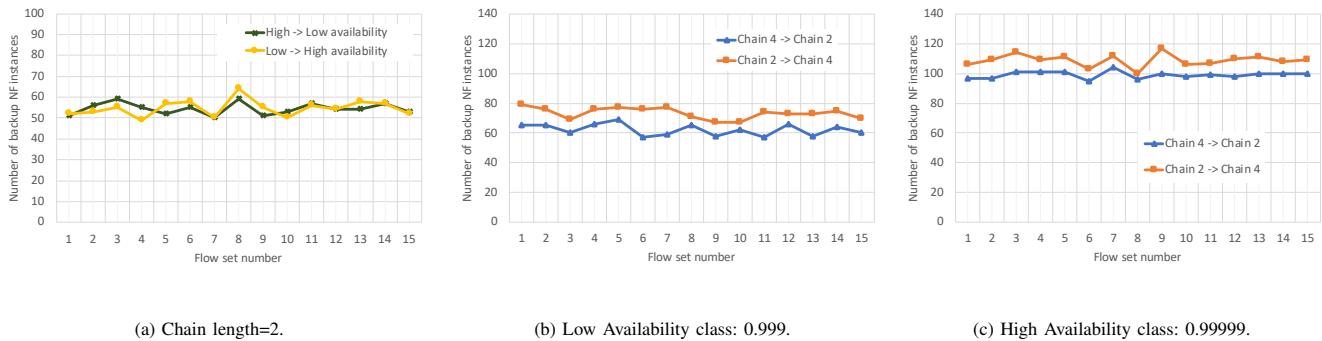
(c) High Availability class: 0.99999.

Fig. 10: Effect of prioritization based on availability requirements (a) and service chain length (b) and (c).

CoShare is able to minimize the impact of correlated failures due to network structural dependence on service availability. In addition, CoShare allows *shared reservation* among flows, i.e. let them share the same reserved backup capacity at an instance, to improve resource efficiency without compromising their availability. This forms another contribution of CoShare. Moreover, CoShare stands out with supporting diverse flow availability requirements under heterogeneous nodes and instances in terms of both resources and availability. The experimental results demonstrate that a redundancy approach without considering network structural dependence in its design can unfortunately fail to meet its promised availability. In addition, when backup capacity is dedicatedly reserved for each flow at a shared NF instance, the performance of CoShare (with dedicated reservation) is close to that of the optimal solution, but CoShare is scalable. Furthermore, with shared reservation, CoShare can reduce the resource overbuild significantly, e.g. about half or more in the Rocketfuel topology experiment. These results indicate that CoShare is appealing for redundancy allocation in NFV. They also imply the criticality and potential of taking into account network structural dependence in addressing the NFV redundancy allocation problem.

References

[1] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[2] B. Blanco *et al.*, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[3] A. Mohammadkhan, K. K. Ramakrishnan, and V. A. Jain, "CleanG—Improving the Architecture and Protocols for Future Cellular Networks With NFV," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, p. 2559–2572, Dec. 2020.

[4] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[5] B. Tola, G. Nencioni, and B. E. Helvik, "Network-aware availability modeling of an end-to-end NFV-enabled service," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1389–1403, 2019.

[6] A. Kamisiński *et al.*, *Resilient NFV Technology and Solutions*. Cham: Springer International Publishing, 2020, pp. 675–697.

[7] N.-T. Dinh and Y. Kim, "An efficient availability guaranteed deployment scheme for IoT service chains over fog-core cloud networks," *Sensors*, vol. 18, no. 11, p. 3970, 2018.

[8] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *25th International Symposium on Quality of Service (IWQoS)*, 2017.

[9] VMware vSphere Availability. Accessed 2020 Jun. [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.7/vsphere-esxi-vcenter-server-671-availability-guide.pdf

[10] S. G. Kulkarni, G. Liu, K. K. Ramakrishnan, M. Arumaithurai, T. Wood, and X. Fu, "Reinforce: Achieving efficient failure resiliency for network function virtualization-based services," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 695–708, 2020.

[11] Y. T. Woldeyohannes and Y. Jiang, "Measures for network structural dependency analysis," *IEEE Communications Letters*, vol. 22, no. 10, pp. 2052 – 2055, 2018.

[12] M. D. Schirmer *et al.*, "Network structural dependency in the human connectome across the life-span," *Network Neuroscience*, 2019.

[13] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning," *Networks*, vol. 70, no. 4, pp. 373–387, 2017.

[14] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *IEEE NFV-SDN*, 2016.

[15] J. Fan *et al.*, "A framework for provisioning availability of NFV in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2246–2259, 2018.

[16] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2759–2772, 2017.

[17] J. Fan *et al.*, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015, pp. 13–18.

[18] S. Herker *et al.*, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *IEEE Globecom Workshops*, 2015.

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

[20] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Designing optimal middlebox recovery schemes with performance guarantees," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2373–2383, 2018.

[21] F. He, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes with importance," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1742–1755, 2019.

[22] A. Tomassilli *et al.*, "Resource requirements for reliable service function chaining," in *IEEE ICC*, 2018.

[23] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 558–573, 2018.

[24] D. Li *et al.*, "Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[25] 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 *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[26] D. Y. Kenett *et al.*, "Dependency network and node influence: Application to the study of financial markets," *International Journal of Bifurcation and Chaos*, vol. 22, no. 07, p. 1250181, 2012.

[27] J. Doucette, M. Clouqueur, and W. D. Grover, "On the availability and capacity requirements of shared backup path-protected mesh networks," *Optical Networks Magazine*, vol. 4, no. 6, pp. 29–44, 2003.

[28] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks. Part I-protection," in *IEEE INFOCOM*, 1999.

[29] G. Li *et al.*, "Efficient distributed path selection for shared restoration connections," in *IEEE INFOCOM*, 2002.

[30] ——, "Efficient distributed restoration path selection for shared mesh restoration," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 761–771, 2003.

[31] C. Ou *et al.*, "New and improved approaches for shared-path protection in WDM mesh networks," *Journal of Lightwave Technology*, vol. 22, no. 5, pp. 1223–1232, 2004.

[32] M. Tornatore *et al.*, "PHOTO: an efficient shared-path-protection strategy based on connection-holding-time awareness," *Journal of Lightwave Technology*, vol. 23, no. 10, p. 3138, 2005.

[33] D. Xu *et al.*, "On the complexity of and algorithms for finding the shortest path with a disjoint counterpart," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 147–158, 2006.

[34] W. Zhang *et al.*, "OpenNetVM: A platform for high performance network service chains," in *Proc. the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016.

[35] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.

[36] I. N. ETSI, "ETSI GS NFV-REL 001 V1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.

[37] J. Zhang *et al.*, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE INFOCOM*, 2019, pp. 1918–1926.

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

[39] Y. T. Woldeyohannes, B. Tola, and Y. Jiang, "Towards carrier-grade service provisioning in NFV," in *15th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2019.

[40] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," *European Journal of Operational Research*, vol. 238, no. 1, pp. 1–17, 2014.

[41] A. Hmaity *et al.*, "Virtual network function placement for resilient service chain provisioning," in *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016, pp. 245–252.

[42] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social networks*, vol. 11, no. 1, pp. 1–37, 1989.

[43] S. Martello and P. Toth, "Bin-packing problem," in *Knapsack problems: Algorithms and computer implementations*. Wiley, 1990, pp. 221–245.

[44] M. Casazza, M. Bouet, and S. Secci, "Availability-driven NFV orchestration," *Computer Networks*, vol. 155, pp. 47–61, 2019.

[45] Y. Bejerano *et al.*, "Algorithms for computing QoS paths with restoration," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 648–661, 2005.

[46] *GEANT – the pan-european research and education network,*, 2019 (accessed December 1, 2019). [Online]. Available: http://www.geant.net.

[47] A.-L. Barabási and E. Bonabeau, "Scale-free networks," *Scientific american*, vol. 288, no. 5, pp. 60–69, 2003.

[48] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[49] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

Besmir Tola received the M.Sc. degree in Electronics and Telecommunication Engineering from the University of Siena, Italy, and the Ph.D. degree from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. From spring 2020, he holds an Assistant Professor position within the department of Information Security and Communication Technology at NTNU. In 2016 and 2018, he was a visiting researcher at the Nokia Bell Labs in Stuttgart (Germany), and UNINETT (Norwegian National Research and Education Network Operator), respectively, where he worked on dependability modeling and analysis of cloud computing infrastructures and services. His research interests include performance and dependability analysis of cloud computing, SDN, and NFV architectures.

Yuming Jiang received the B.Sc. degree from Peking University and the Ph.D. degree from the National University of Singapore. He has been a Professor with the Norwegian University of Science and Technology, Trondheim, Norway, since 2005. From 1996 to 1997, he was with Motorola, Beijing, China, and from 2001 to 2003, he was with the Institute for Infocomm Research (I2R), Singapore. His research interests are the provision, analysis, and management of quality of service guarantees. He has authored the book entitled Stochastic Network Calculus. He was a Co-Chair of IEEE Globecom 2005 - General Conference Symposium, a TPC Co-Chair of 67th IEEE Vehicular Technology Conference (VTC) 2008, the General Chair of IFIP Networking 2014 Conference, the Chair of the 2018 International Workshop on Network Calculus and Applications, and a TPC Co-Chair of the 32nd International Teletraffic Congress (ITC32), 2020.

Dr. K. K. Ramakrishnan is Professor of Computer Science and Engineering at the University of California, Riverside. Previously, he was a Distinguished Member of Technical Staff at ATT Labs-Research. Prior to 1994, he was a Technical Director and Consulting Engineer in Networking at Digital Equipment Corporation. Between 2000 and 2002, he was at TeraOptic Networks, Inc., as Founder and Vice President. K. K. is an ACM Fellow, an IEEE Fellow and an ATT Fellow, recognized for his fundamental contributions on communication networks, including his work on congestion control, traffic management and VPN services. He has published over 300 papers and has 185 patents issued in his name. K. K. received his MTech from the Indian Institute of Science (1978), MS (1981) and Ph.D. (1983) in Computer Science from the University of Maryland, College Park, USA.

Yordanos Tibebu Woldeyohannes received the B.S. degree in Electrical Engineering from Mekelle University, Ethiopia in 2007, the M.S degree in Telecommunication Engineering (cum laude) from University of Trento in 2013, Italy and the PhD degree in Information Security and Communication Technology from Norwegian University of Science and Technology in 2020. From 2008 to 2010 she was with ZTE corporation Ethiopian branch. From 2019, she is with Gintel AS. Her research interests include mathematical modeling and algorithm design (optimization), resource allocation, network function virtualization and software defined networks.