Towards Carrier-Grade Service Provisioning in NFV

Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang
NTNU-Norwegian University of Science and Technology, Norway

***Abstract*—Network Function Virtualization (NFV) is an emerging technology that reduces cost and brings flexibility in the provisioning of services. NFV-based networks are expected to be able to provide carrier-grade services, which require high availability. One of the challenges for achieving high availability is that the commodity servers used in NFV are more error prone than the purpose-built hardware. The "de-facto" technique for fault tolerance is redundancy. However, unless planned carefully, structural dependencies among network nodes could result in correlated node unavailabilities that undermine the effect of redundancy. In this paper, we address the challenge of developing a redundancy resource allocation scheme that takes into account correlated unavailabilities caused by network structural dependencies. The proposed scheme consist of two parts. In the *first part*, we propose an algorithm to identify nodes that can be highly affected by a node failure because of their network structural dependency with this node. The algorithm analyzes such dependencies using a recently proposed centrality measure called *dependency index*. In the *second part*, a redundancy resource allocation scheme that places backup network functions on nodes considering their dependency nature and assigns the instances to flows optimally is proposed. The results show that not considering the network structural dependency in backup placement may significantly affect the service availability to flows. The results also give insights into the trade-off between cost and performance.**

I. Introduction

Middleboxes or Network Functions (NFs) are widely utilized for various purposes such as improving security and network performance. The traditional approach of having a dedicated purpose-built hardware per NF has been shown to be inefficient and expensive [1]. Network Function Virtualization (NFV) alters this inflexible architecture by decoupling the software of NFs from the hardware and run the NFs on virtualized environment such as virtual machines (VMs) or containers. NF instances can then be created on the fly depending on the traffic and the network state [2]. The VMs and containers of the NFs are usually hosted on commercial off-the-shelf (COTS) hardware, which are comparatively less expensive than the purpose-built hardware. A service in NFV is typically composed of a set of NFs that are chained in some specific order, also known as service function chaining.

Carrier-grade services such as telecommunication services require high-level of availability reaching five-nines (99.999%) or more [3] [4]. Achieving this level of availability in NFV networks is challenging due to a number of reasons including: lower dependability of COTS servers, correlated failures or unavailabilities, and state management:

COTS availability: Legacy telecommunication networks have achieved carrier-grade service availability by using purpose-built hardware. In NFV, the purpose-built hardware is replaced by COTS hardware, which is usually more error-prone [4]. To achieve the same level of availability using COTS, NFV needs to build the resilience into software [5].

Correlated failures / unavailabilities: Although most literatures assume that failures are independent, correlated failures or unavailabilities are often common in real systems [6], [7]. For example, the failure of a node may result in the unavailability of other nodes because of their structural dependencies on this node [8]. The de-facto technique for boosting availability is redundancy. However, redundancy may become ineffective due to correlated failures or unavailabilities.

State management: A large number of NFs such as Deep Packet Inspection (DPI) and Network Address Translator (NAT) are stateful. Stateful NFs preserve service states, such as, TCP connection state and the mapping between IP addresses about ongoing connections [9]. Typically, NFs need to maintain 10-100s of state variables that are per-flow or shared across flows [10]. Backup instances of stateful NFs need to have updated state information to ensure successful failover and service continuity [11], [12].

In this paper, we make a step forward towards carrier-grade service provisioning in NFV, by proposing a novel redundancy resource allocation scheme where two crucial challenges are addressed. (1) One challenge is *how to factor the inherent network structural dependency among nodes into redundancy resource allocation*. (2) The other challenge is *how to efficiently place and allocate backup instances for service chain of flows*.

To tackle the first challenge, an algorithm that measures the dependency among network nodes and identifies nodes that have a high-level of structural correlation is proposed. The dependency among nodes of a network is quantified by using a centrality measure called *node dependency index* [13]. Here, there is an intuition, which is, a backup NF should not be placed at a node that may also become unavailable when the primary NF's node fails. For the second challenge, an optimization model that aims to efficiently place redundant NFs and assign backup NFs to service chains of flows is proposed. In this model, flows are assigned backup instances following the $1 : m$ active-standby redundancy mode, with which, every flow can tolerate the failure of any one of the NF instances on the NF chain [11]. In addition, following the intuition, redundant instances are not placed on backup nodes that are structurally correlated with the primary nodes. Furthermore, to efficiently utilize resources, backup NFs are shared by flows and only the required number of instances are created.

Most of the existing works on redundancy allocation in NFV based networks focus on providing two-nines or three-nines service availability [14], [15]. only a few consider carrier-grade service availability [16], [17]. Both [16] and [17] allocate only on-site redundancies. However, to guarantee carrier-grade service availability it is important to also have backups distributed geographically [4]. In addition, [17] considers only the failure of the physical nodes while not the NF applications and assumes that all nodes have the same availability, and [16] focuses on the failure of VMs assuming similar availability and failure independence between VMs.

Our proposed scheme differs from the existing works in a number of ways. First, our scheme considers the effect of network structural dependency. Second, it takes into account both physical hardware failures and NF software failures with different availability values. Third, it can be used to allocate both on-site and off-site backups in an optimal way. Moreover, in our proposed scheme, in addition to availability, delay performance is also taken into consideration, such that the delay that flows experience after failover can be kept within the requirement of the flows.

The specific contributions in this paper include:

- An algorithm that identifies the set of nodes that have strong structural correlation using a centrality measure called node dependency index.
- A redundancy allocation scheme that finds the optimal number and placement of backup nodes and NF instances and assigns the instances to flows.

The paper is organized as follows. In Section II, the system model is described. Section III discusses in brief the node dependency index centrality measure. In Section IV, the algorithm proposed for identifying the nodes that have high-level of structural correlation is explained. The proposed redundancy allocation scheme is presented in Section V, followed by the results in Section VI. Finally, Section VII presents the concluding remarks.

II. System Model

The system considered is a network of nodes and links and is represented as a graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ represents the set of links. Nodes hosting NFs that are being utilized by the primary service chains of flows are called *primary nodes*. Nodes that can be used for backup are called *backup nodes*. $\mathcal{B}$ denotes the set of backup nodes and $\mathcal{P}$ the set of primary nodes. Backup NFs are hosted on backup nodes.

A node hosting backup instances can be a shared or dedicated backup node. A shared backup node is a node that is being used both as a primary and backup node. This type of nodes reserve some resources to be used as backup while hosting NFs that are utilized by the primary service chains. Dedicated backup nodes are nodes that are used only to host backup instances.

Each flow f has a source and destination node pair, which are respectively denoted as s_f and d_f. The service required by flow f is represented by a service chain, $\overrightarrow{S}_f = (S_f^1, S_f^2 \ldots S_f^{g_f})$. The service chain is an ordered series of network functions, where S_f^1 is the first NF, S_f^2 is the second NF needed and so on. It is assumed that a flow is already assigned a primary service chain. The variable $p_{f,g}$ indicates the primary node p that is serving flow f's g^{th} service. A backup instance of a type v NF requires k_v number of cores and can be a backup to up to C_v^b number of flows, where b is backup host node of NF type v instance. For each flow f, there is an availability requirement on its service $\overrightarrow{S}_f$, denoted as A_f. A service $\overrightarrow{S}_f$ is considered available if either the primary or one of the backup service chains is available.

III. Structural Dependency Measures

The *node dependency index* $DI(i|n)$ measures the average level of dependency node i has on node n in connecting with the other nodes of the network [13]. $DI(i|n)$ is calculated from the path dependency index $DI(i \to j|n)$, which measures the dependency the path between nodes i and j has on node n. $DI(i \to j|n)$ is defined as:

$$DI(i \to j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } A_{ij}^{-n} = 1 \\ 1 & \text{if } A_{ij}^{-n} = 0, \end{cases} \tag{1}$$

where I_{ij} is an information measure, which is equal to the inverse of the shortest path distance hop counts, denoted as d_{ij}, between nodes i and j, i.e. $I_{ij} = 1/d_{ij}$. I_{ij}^{-n} is the information measure between nodes i and j after the deactivation of node n. The binary variable A_{ij}^{-n} measures the availability: $A_{ij}^{-n} = 1$ if node i can reach node j after the deactivation of node n and zero otherwise. The node dependency index is defined as:

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

$DI(i|n)$ measures the average dependency that node i has on node n. $DI(i|n) = 1$ if node i cannot connect with the other nodes, $DI(i|n) = 0$ if node i does not experience any connectivity problem and $0 < DI(i|n) < 1$, if node i experiences connectivity problem but is still able to connect to at least one other node, after the failure of node n.

IV. Structurally Correlated Nodes

While failure independence is commonly assumed when studying availability, recent studies have demonstrated the existence of correlated failures and the pronounced effect of geographical adjacency [7] [18], [19]. Nevertheless, it has also been recognized that it is difficult to discover or predict dependencies among failures [6], [7] [18], [19]. To tackle this challenge, in this section, a novel approach is proposed to identify nodes that are inherently correlated due to the network structure. This information lays a foundation for the proposed redundancy allocation scheme that will be detailed in Section V.

A. Algorithm

The failure of a node may result in the unavailability of other nodes. For example, in a data-center network, the failure of a Top-of-Rack switch will result in the unavailability of all the servers located in the same rack. The proposed algorithm uses the node dependency index to measure the dependency among nodes and identify the nodes that have high structural correlation. From the definition of the node dependency index, if node i has high-level of dependency on node n, the failure of node n might result in the unavailability of node i.

Definition 1: *Critical nodes of node* i, denoted as $\mathcal{C}(i)$, are nodes that node i is highly dependent on, where node i is said to be highly dependent on node n if $DI(i|n)$ is above a given threshold t_{DI},

$$\mathcal{C}(i) = \{n | DI(i|n) > t_{DI}, n \in \mathcal{N}\}. \tag{3}$$

If $\mathcal{C}(i)$ is empty, node i is independent of the other nodes of the network so has no critical node. For example, in a fully mesh network, all nodes are independent of each other as the failure of one does not affect the connectivity of the others.

To find the set of nodes that have strong structural correlation with a primary node i, some intuitive observations are used which include:

First-level dependency

- Node i has a high probability of experiencing a correlated failure with its critical nodes in $\mathcal{C}(i)$ as the failure of these nodes might lead to the unavailability of node i. Thus, node i should not use the nodes in $\mathcal{C}(i)$ as a backup.
- Node i should not also use as a backup those nodes that depend on it highly. Since the failure of node i might also result in the unavailability of these nodes.

In brief, *a primary node* i *and its backup nodes should not depend on each other.* This can be regarded as the *first-level dependency* among nodes.

Second-level dependency

- Node i should not use as a backup nodes that depend heavily on its critical node. This is because if the unavailability of node i is due to the failure of its critical node, the other nodes that depend heavily on the critical node might also be unavailable.

The algorithm for finding a set of nodes, $\hat{\mathcal{B}}_i$, which are structurally correlated with a primary node i is shown in Algorithm 1. The algorithm starts by finding the critical nodes of a primary node. The critical nodes of node i, $\mathcal{C}(i)$, are inserted into the set $\hat{\mathcal{B}}_i$. Then, nodes that have a high-level of dependency on node i are included to the set $\hat{\mathcal{B}}_i$ (line 9). For the second-level dependency, all the nodes that are highly dependent on the critical nodes of node i will be included to $\hat{\mathcal{B}}_i$. The threshold, t_{DI}, should be assigned values that are between zero and one. If it is set to a very low value that is close to zero, the set $\mathcal{C}(i)$ will include a large number of network nodes. Therefore, it should be assigned a relatively large or medium values such as 0.5.

Algorithm 1 Heuristic for finding structurally correlated nodes

```
1:  $G(\mathcal{N}, \mathcal{L}) \rightarrow$ the network graph.
2:  $\hat{\mathcal{B}}_i \rightarrow$ set of nodes that are structurally correlated with
    node $i$.
3:  $t_{DI} \rightarrow$ threshold for high dependency
4:  for  $i \in \mathcal{N}$ do
5:      Find $\mathcal{C}(i)$ using $t_{DI}$
6:      Insert $\mathcal{C}(i)$ to $\hat{\mathcal{B}}_i$
7:      for  $j \in \mathcal{N} \ /i$ do
8:          if $i \in \mathcal{C}(j)$ then
9:              Insert $j$ to $\hat{\mathcal{B}}_i$
10:         if $j \in \mathcal{C}(i)$ then
11:             for  $k \in \mathcal{N} \ /i$ do
12:                 if $j \in \mathcal{C}(k)$ then
13:                     Insert $k$ to $\hat{\mathcal{B}}_i$
14: return $\hat{\mathcal{B}}_i$
```

V. Redundancy Allocation Scheme

Some of the features considered in the design of the redundancy allocation scheme include:

- *Correlated failures*: To tolerate correlated failures caused by network structural dependency, backup NFs of a flow are not placed on nodes that are structurally correlated with the primary nodes of the flow.
- *State:* Stateful NFs can have states that are per-flow or shared across flows. Flows using the same primary stateful NF instance will be assigned to the same backup instance since they rely on a state shared among them.
- *Delay vs utilization:* To efficiently use network resources, minimal number of backup instances are created. However, this can increase the end-to-end backup chain delay of flows. To solve this problem, the scheme finds a balance between minimizing the backup chain delay, which is the delay flows will experience after failover, and the resource utilization.

A. Formulation: All-One

The first model considered is the All-One model in which *all* the services of a chain are assigned one backup that is a 1:1 active-standby mode. It is assumed that *one* backup node will be used to backup all the NFs of a flow, later on this assumption will be relaxed. The redundancy allocation is then formulated as an Integer Linear Program (ILP).

The redundancy allocation has three main objectives: (I) *to minimize the number of backup instances created*, (equation (4)), (II) *to minimize the number of backup nodes used*, (equation (5)), and (III) *to minimize the backup chain delay*, (equation (6)).

$$\text{minimize} \quad \sum_{\forall b \forall v} z_v^b \tag{4}$$

$$\text{minimize} \quad \sum_{\forall b} q^b \tag{5}$$

$$\text{minimize} \quad \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b) \tag{6}$$

The weighted sum method is used to combine the three objective functions into one by using equal unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [20]. The All-One optimization model is given us:

All-One:

$$\min. \sum_{\forall b \forall v} z_v^b + \sum_{\forall b} q^b + \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b). \quad (7)$$

s.t.

$$1 - (1 - \sum_{\forall b} i_f^b A^b \prod_{g=1, v=S_f^g}^{g_f} A_v)(1 - A_{s_f}^p) \geq A_f \quad , \forall f : A_{s_f}^p < A_f \quad (8a)$$

$$\sum_{\forall f, \forall g / S_f^g = v} y_{f,g}^b \leq C_v^b \quad , \forall b, v \quad (8b)$$

$$\sum_{\forall v} z_v^b k_v \leq K^b \quad , \forall b \quad (8c)$$

$$y_{f,g}^b = 0, \quad , \forall f, g \in \{1..g_f\}, \forall b \in \hat{\mathcal{B}}_p / p \in \mathcal{P}_f \quad (8d)$$

$$y_{f,g}^b = y_{f',g'}^b \quad , \forall f, f' \in \mathcal{F} / p_{f,g} = p_{f',g'}, S_f^g = S_{f'}^{g'}, T(S_f^g) = 1, \forall b, g, g' \quad (8e)$$

$$\sum_{\forall b} y_{f,g}^b = 1 \quad , \forall f : A_{s_f}^p < A_f, g \in \{1..g_f\} \quad (8f)$$

$$\sum_{\forall b} i_f^b = 1 \quad , \forall f : A_{s_f}^p < A_f \quad (8g)$$

$$y_{f,g}^b \leq z_v^b \quad , \forall b, f, g \in \{1..g_f\}, v = S_f^g \quad (8h)$$

$$q^b = \max_{v \in \mathcal{V}} z_v^b \quad , \forall b \quad (8i)$$

$$i_f^b = y_{f,g}^b \quad , \forall f, b, g \in \{1..g_f\} \quad (8j)$$

The constraints are classified into five group, which are *availability*, *capacity*, *correlated failure*, *state* and *assignment* constraints. Constraint (8a) belongs to the *availability* group and ensures that flows' availability requirement is fulfilled by the primary and backup NFs assigned. Constraints (8b) and (8c) are *capacity* constraints for the backup NF instances and backup nodes respectively. For each flow, constraint (8d) prohibits the usage of backup nodes that have high structural *correlation* with the primary nodes of the flow. Constraint (8e) is a *state* constraint, which makes sure that flows using the same primary instance of a stateful NF are assigned to the same backup instance.

The other constraints belong to the *assignment* group. *All* the services of a flow have a backup (constraint (8f)) and only *one* backup node hosts all the backup NFs of a flow (constraint (8g)). A flow is mapped to a backup instance on a given backup node only if the node is hosting the NF type (Constraint (8h)). Constraint (8i) identifies backup nodes that are hosting instances. A flow is assigned to a backup node only if it is using backup instance hosted on the node (Constraint (8j)).

TABLE I: Symbols used in formulation

Notation	**Meaning**
K^b	the number of cores available on backup node b.
$D(p, b)$	the delay between nodes p and b.
A^b	the probability that backup node b is available.
$\hat{\mathcal{B}}_p$	set of backup nodes that have high structural correlation with node p.
k_v	the number of cores needed to instantiate NF type v.
$T(v)$	binary variable to show if NF type v is stateful ($T(v) = 1$) or not ($T(v) = 0$).
C_v^b	the maximum number of flows that an instance of NF type v hosted on node b can be a backup to.
A_v	the probability that the application software of a network function of type v is available.
$\mathcal{F}$	the set of flows.
A_f	the availability requirement of flow f.
s_f, d_f	source and destination nodes of flow f respectively.
$\overrightarrow{S}_f = (S_f^1, S_f^2 \dots S_f^{g_f})$	service chain of flow f.
A_{sf}^p	the availability of the primary service chain of flow f.
$\mathcal{P}_f$	the set of primary nodes used by flow f.
$p_{f,g}$	primary node p used by flow f's g^{th} service.
Decision variables	
$y_{f,g}^b$	a binary decision variable, to indicate if backup node b is used as a backup for flow f's g^{th} service.
z_v^b	an integer decision variable to indicate the number of backup instances of NF type v hosted on backup node b.
i_f^b	a binary variable that indicates if backup node b is used by flow f or not.
q^b	a binary variable to indicate if backup node b is hosting backup NF instances.

B. Formulation: All-Any

The "All-One" ILP model given above uses one backup node to backup all the NFs of a flow. This constraint is relaxed so that a flow can use one or more backup nodes. This model will be referred as "All-Any" since *all* of the services of a flow are backed up and a flow can use *any* number of backup nodes. The objective function for minimizing the backup chain delay needs to be modified as

All-Any:

$$\text{minimize} \sum_{\forall b \forall f} (D(s_f, b) y_{f,1}^b + D(b, d_f) y_{f,g_f}^b + \sum_{\forall b' \in \mathcal{B}, g=1}^{g_f - 1} D(b, b') y_{f,g}^b y_{f,g+1}^{b'}). \quad (9)$$

Since a flow might use more than one backup node, the backup chain delay will include the delay between the backup nodes. All the constraints except three (8a, 8g and 8j) of the All-One model will also be included in the All-Any model. The three constraints will be replaced by constraints (10, 11 and 12) respectively.

$$1-(1-\prod_{\forall b}\max(1-i_f^b, i_f^b A^b \prod_{g=1}^{g_k}\max(1-y_{f,g}^b, y_{f,g}^b A_v)))$$
$$(1-A_{s_f}^p)\geq A_f \qquad \forall f: A_{s_f}^p < A_f \quad (10)$$

$$\sum_{\forall b} i_f^b \geq 1 \qquad \forall f: A_{s_f}^p < A_f \qquad (11)$$

$$i_f^b = \max_g(y_{f,g}^b) \qquad \forall b, \forall f: A_{s_f}^p < A_f \qquad (12)$$

Constraint (10) guarantees that the availability requirement of flows is satisfied by the primary and backup instances, which might be hosted on different backup nodes. One or more backup nodes are assigned to a flow (constraints (11)). A flow is assigned a backup node provided that it is using atleast one backup instance hosted on the node (constraint (12)). This model is an Integer Non-linear Program (INLP) because of the non linearity of equation (10). To decrease the complexity of the model, the non-linear constraint is approximated by a linear equation.

1) Linear approximation: The availability constraint is approximated by a linear lower bound function.

Theorem 1. *The probability that all E entities of a set $\mathcal{E}$ will be available is lower bounded by $1-\sum_{e=1}^{E}U_e$, where every entity $e\in\mathcal{E}$ fails independently with probability A_e and U_e is the unavailability of entity e.*

Proof: The probability that all the E entities will be available (A_t) is a product of the availability of each of them. That is

$$A_t = \prod_{e=1}^{E} A_e. \qquad (13)$$

By definition, the availability of entity e, $A_e = 1-U_e$, where U_e is the unavailability of e. Thus,

$$A_t = \prod_{e=1}^{E}(1-U_e). \qquad (14)$$

By expanding the product,

$$\prod_{e=1}^{E}(1-U_e) = 1-\sum_{e=1}^{E}U_e + \sum_{e=1}^{E-1}U_eU_{e+1} + o(n), \qquad (15)$$

where $o(n)$ represents the higher order terms. Usually, unavailability $U << 1$ so the product and the higher order terms can be ignored. We will then have

$$\prod_{e=1}^{E}(1-U_e) \geq 1-\sum_{i=1}^{E}U_e. \qquad (16)$$

As a result,

$$A_t \geq 1-\sum_{e=1}^{E}U_e, \qquad (17)$$

which concludes the proof. ■

For example, if there are two entities with availability $A_1 = 0.9$ and $A_2 = 0.99$, then $A_t = 0.891$. Using the linear approximation, $U_1 = 0.1, U_2 = 0.01$, so $A_t = 0.89$. Thus, the linear approximation is a lower bound to the actual availability value. Applying this linear approximation, equation (10) can be approximated by:

$$1-(1-(1-\sum_{\forall b} i_f^b(U^b + \sum_{g=1}^{g_k} y_{f,g}^b U_v)))(1-A_{s_f}^p)$$
$$\geq A_f \qquad ,\forall f: A_{s_f}^p < A_f, \qquad (18)$$

where U^b and U_v are the unavailabilities of backup node b and backup NF v respectively. The lower bound approximation is conservative so flows availability requirement will not be violated. Equation (18) is not linear since it has a term that is a product of two variables, i_f^b and $y_{f,g}^b$. Let variable $r_{f,g}^b = i_f^b y_{f,g}^b$,

$$1-(1-(1-\sum_{\forall b} i_f^b U^b - \sum_{\forall b}\sum_{g=1}^{g_k} r_{f,g}^b U_v))(1-A_{s_f}^p)$$
$$\geq A_f \qquad ,\forall f: A_{s_f}^p < A_f. \qquad (19)$$

Equation (19) is a linear equation of the variables i_f^b and $r_{f,g}^b$. Thus, the non-linear inequality constraint in equation (10), can be substituted by equation (19) and the constraint $r_{f,g}^b = i_f^b y_{f,g}^b$. However, since the variables i_f^b and $y_{f,g}^b$ are binary, their product can easily be linearized by substituting it with the following linear equations,

$$r_{f,g}^b <= i_f^b$$
$$r_{f,g}^b <= y_{f,g}^b$$
$$r_{f,g}^b >= i_f^b + y_{f,g}^b - 1. \qquad (20)$$

Thus, the equivalent ILP model of the All-Any model will have constraints (19) and (20) instead of the non-linear availability constraint and all the other linear constraints of the All-Any INLP model.

C. Allocating more than one backup chain

The All-One and All-Any models assign one backup for each of the NFs of a flow's service chain. However, to guarantee the high availability of carrier-grade services, it might be necessary to allocate more than one backup chain. The following simple example is used to showcase this. Consider a flow that has two services long chain. The primary chain of the flow is 90% available and the flow requires to be 99.999% available. The backup nodes and NF applications are 99% and 99.9% available respectively. Thus, after being allocated backup instances that are hosted on the same backup node, the flow will only be 99.88% (2'9s) available. Thus, a second backup chain is needed to reach the required 99.999% (5'9s) availability.

Algorithm for assigning more than one backup chain: Backup instances are to be allocated for a set ($\mathcal{F}$) of flows. The proposed models assign one backup chain. More than one backup chains are allocated to a flow sequentially one after the other. That is the first backup chain is allocated and

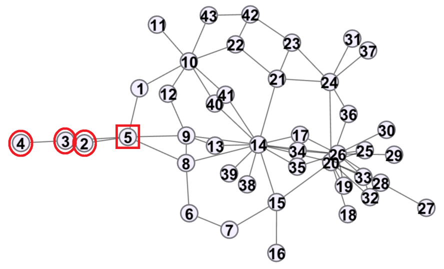


Fig. 1: GEANT network: Example of structurally correlated nodes

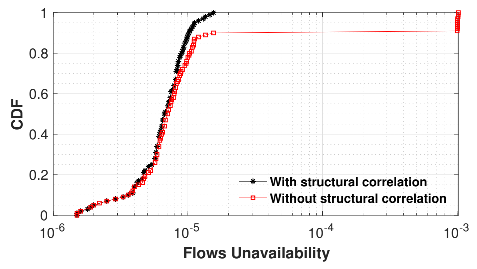


Fig. 2: Effect of not considering structural correlation: CDF of the unavailability

if the availability requirement of the flow is not satisfied then the second backup chain is assigned and so on. One problem with using the models directly for assigning backup chains sequentially is that if the availability requirement of a flow $f \in \mathcal{F}$ cannot be satisfied by one backup chain, the models will be infeasible. To solve this issue, for all of the flows in the set, it is checked whether their availability requirement can be satisfied while being assigned to the least available backup node. If not, the availability requirement of the flow will be downgraded to the next availability class, e.g., from 99.999% to 99.99% or from 99.99% to 99.9%. The original availability requirement of the flow is saved and the flow will be marked as a flow that might need more than one backup.

Then, the first backup chain will be allocated by using the models. The state of the network (including the placement of backup instances and their capacity) will then be updated. If the availability requirement of a flow is not satisfied by the first backup chain, then the same process will be used to allocate the second backup chain. Two variables are introduced to transfer the state of the network between the different rounds of backup chain allocations. These are oZ_v^b, the number of backup instances of type v already created on node b, and oC_v^b, the currently available capacity of an existing instance of type v hosted on node b. For this algorithm, in the models z_v^b will be replaced by $z_v^b + oZ_v^b$ and C_v^b will be replaced by $C_v^b + oC_v^b$. This is done to be able to use instances created previously in the current round of the backup allocation.

In case it is not possible to allocate backups due to shortage of resources, flows will be rejected. Resource shortage can occur at any round of the backup chains assignment. For example, when a flow is allocated a second backup chain. In this case, the flow has already been assigned one backup chain. However, the availability requirement of the flow is not yet satisfied. In cases like this, the flow will be rejected and the resources already assigned to it will be released to be used by other flows.

VI. Results

The performance of the proposed scheme is analyzed by conducting a number of experiments. The models are solved by using a commercial solver, CPLEX, together with Matlab for transferring updated network state information between different rounds. The GEANT network, Fig. 1, which is the pan-European research and education network, is used as a test network topology [21].

Eight nodes of the network are chosen to be the ingress/egress nodes. The ingress/egress nodes are a bottleneck for achieving high availability since the failure of one of them leads to service unavailability for customers using it. To avoid this, these nodes are paired to provide "dual homing", whereby one node is a backup for the other and vice versa. Twenty-two nodes of the network are assumed to be backup nodes (in shared or dedicated mode). Each backup node has 4 CPU cores to be used by the backup instances it hosts. The rest of the nodes are dedicated primary nodes. The availability of the nodes is assumed to be uniformly distributed between $0.99 - 0.999$, whereas NF instances have an availability between $0.999 - 0.9999$ and an NF instance can be a backup for up to 10 flows.

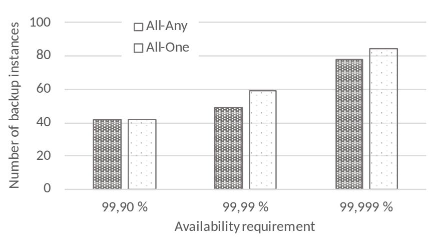
Flows are assumed to require a service chain that is composed of two NFs. NFs of a chain are randomly chosen out of the set of five services, which are Firewall, DPI, IDS, Proxy, NAT. Flows are assigned primary chains by using ClusPR algorithm [2]. The availability requirement of a flow is selected from the set $\{0.999, 0.9999, 0.99999\}$.

A. *Structurally correlated nodes*

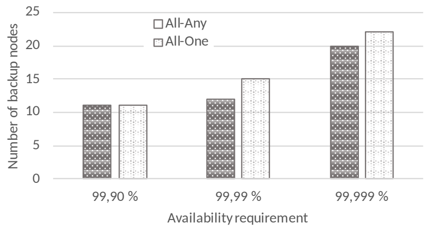
Example of nodes that have high structural correlation, which are identified by the proposed algorithm are highlighted in Fig. 1. Nodes 2-4 have a high probability of experiencing a correlated failure with node 5 due to their dependencies. This is because, the failure of node 5 will lead to the unavailability of these nodes as well.

1) Effect of not considering structural correlation: In this section, a simple experiment is carried out to showcase the effect of not considering the structural correlation among nodes in the backup instance placement decision making. The baseline algorithm from [16] is used to decide the number of backup instances needed. It is assumed that the availability of a node is 0.999 (3'9s). According to the baseline algorithm, theoretically, one backup for each of the NFs is enough to meet the 99.999% availability requirement of a single service function chain containing two NFs. The primary and backup NF host nodes of a chain are randomly chosen from the network. When structural correlation is considered, the backup nodes of a chain will not have strong correlation with the primary nodes.

The availability of 100 flows is 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

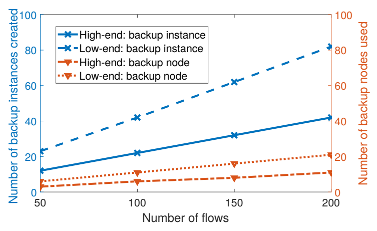


(a) Backup instances

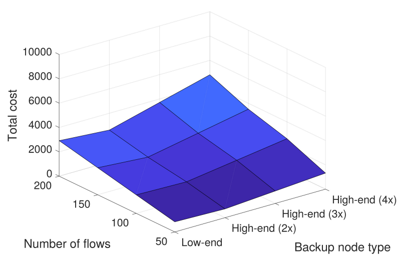


(b) Backup nodes

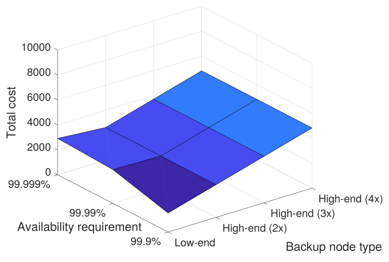
Fig. 3: Number of backup NF instances and nodes utilized for acheiving different availability requirements



(a) Backup nodes and instances



(b) Cost for achieving 99.999% availability



(c) Cost for 200 flows

Fig. 4: “High-end” vs “Low-end”: Number of backup NF instances created and backup nodes used (a), total cost for achieving 5’9s (b), total cost for 200 flows (c).

from Bernoulli distribution using the node’s availability. The network is then updated considering the nodes state. Finally, the availability of the backup and primary chains is checked by verifying the availability of the host nodes of the chain’s NFs and the path between them. The chain is said to available if either the primary or backup chain is available. A CDF of the unavailability of the 100 flows is shown in Fig. 2. When structural correlation is not considered around 10% of the flows have low availability, 3’9s and 2’9s.

B. Resource utilization

The number of backup instances created and nodes utilized for fulfilling the availability requirements of 200 flows for different availability requirements are shown in Fig. 3a and 3b respectively.

When the flows have 99.9% availability requirement, 42 backup instances are created and 11 backup nodes are used to host the instances by both All-Any and All-One models. The availability requirement of all of the flows is able to be fulfilled with 1:1 active-standby backup for each of the NFs of a chain. When the availability requirement of the flows increases to 99.99%, 49 backup instances are created by the All-Any model and 59 by the All-One model. For some of the flows, 1:1 active-standby was not enough to reach to the required availability therefore, a 1:2 active-standby, where one NF of a chain has two backup instances, is required. As a result, more backup instances are created. When comparing the two models, the All-One model created more instances than the All-Any model. This is because of the All-One model’s constraint that forces a flow to use backup instances hosted only on the same node. For achieving 5’9s (99.999%) availability, most of the flows need 1:2 backup protection.

C. Effect of the type of backup nodes

In this section, the effect of using highly available COTS servers versus COTS with lower availability, referred to as “High-end” and “Low-end” respectively is analyzed. The “High-end” servers are assumed to be 99.9% available and the “Low-end’ servers 99% available. The relative importance between the cost of installation of host server hardware and the cost of installation of the network function software license is chosen to be 100:10 for “Low-end” servers as in [22]. For the “High-end” servers, the cost is 200:10 if the “High-end” servers are twice ($2\times$) more expensive, and 300:10 if they are $3\times$ more expensive compared to the “Low-end”.

Figure 4a and 4b show the number of backup instances created, nodes utilized and the total cost for fulfilling 99.999% availability requirement of different number of flows. For the “Low-end” servers, 1:2 backup have to be applied to reach the 5’9s requirement. Using the “High-end” servers, the availability requirement is fulfilled with 1:1 active-standby redundancy. Therefore, fewer backup instances are created when using “High-end” servers. However, the “High-end” servers are more expensive than the “Low-end” servers. The total cost spent for fulfilling the availability requirement of the flows depends on the relative cost of the servers. It is

TABLE II: Effect of including delay in the objective function

Objective	Average delay (hops)	Worst-case delay (hops)	# Backup instances
Without delay	4.68	12	12
With delay	0.44	2	15

more economical to use "High-end" servers if their cost is not more than 2× the cost of the "Low-end" servers. If the cost of installation of the "High-end" servers is 3× or more compared to the "Low-end" servers, the total cost spent will be more than that spent using the "Low-end" servers.

Figure 4c shows the total cost for serving 200 flows when their availability requirement changes. The 1:1 active-standby protection is enough for meeting the 99.9% availability requirement. Thus, it is economical to use the "Low-end" servers. For both 99.99% and 99.999%, if the "High-end" servers are 2× more expensive or less, then it is more economical to use the "High-end" servers.

D. Effect of minimizing the backup chain delay

The backup chain delay, in terms of number of hops, observed when the objective includes minimizing the end-to-end delay and the amount of resources used is compared with the case when the objective is only to minimize the resources used (i.e., instances created and nodes utilized).

Table II shows the results of the comparison for 50 flows that have 99.9% availability requirement. When the objective is to minimize the resource utilization, 12 instances are created. Compared to the primary chain, the backup chain delay is 4.68 hops longer on average. In the worst case, a flow's backup chain is 12 hops longer than its primary one. When the objective function is to minimize both the total backup chain delay and the resource utilization, the average backup chain delay is only 0.44 hop counts longer and the worst-case observed delay is 2 hops longer. In this case, 15 backup instances are created.

VII. Conclusion

In this paper, a redundancy resource allocation scheme that tolerates correlated failures caused by network structural dependency is proposed. The scheme identifies the sets of nodes that have strong structural correlation using a novel algorithm that is based on the node dependency index centrality measure. The experimental results demonstrate that not taking into account the structural correlation among nodes in backup instances placement decision making considerably affects the availability of flows. The results also give insights into the trade-off between cost and system performance.

Acknowledgment

The work for this paper was performed in the context of the EU FP7 Marie Curie Actions project Grant Agreement No. 607584 (the CleanSky project). The authors would also like to thank Prof. K. K. Ramakrishnan for his insightful comments.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] 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*, pp. 1–1, 2018.

[3] D. Collins, *Carrier grade voice over IP*. McGraw-Hill New York, 2003, vol. 2.

[4] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 152–157, 2017.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[6] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz, "Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems," in *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*. IEEE, 2002, pp. 362–367.

[7] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Subtleties in tolerating correlated failures in wide-area storage systems." in *NSDI*, vol. 6, 2006, pp. 225–238.

[8] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "Heading off correlated failures through independence-as-a-service." in *OSDI*, 2014, pp. 317–334.

[9] S. Woo, J. Sherry, S. Han, S. Moon, S. Ratnasamy, and S. Shenker, "Elastic scaling of stateful network functions," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 299–312.

[10] J. Khalid, A. Alsudais, E. Keller, and F. Le, "Paving the way for NFV: Simplifying middlebox modifications using statealyzr." in *NSDI*, 2016, pp. 239–253.

[11] N. ISG, "Network function virtualisation (NFV)-resiliency requirements," *ETSI GS NFV-REL*, vol. 1, p. v3, 2016.

[12] P. X. Sherry, Justine Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo *et al.*, "Rollback-recovery for middleboxes," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 227–240.

[13] Y. T. Woldeyohannes and Y. Jiang, "Measures for network structural dependency analysis," *IEEE Communications Letters*, 2018.

[14] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

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

[16] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*. IEEE, 2017, pp. 1–10.

[17] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[18] A. J. Gonzalez, B. E. Helvik, J. K. Hellan, and P. Kuusela, "Analysis of dependencies between failures in the UNINETT IP backbone network," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*. IEEE, 2010, pp. 149–156.

[19] B. Chun and A. Vahdat, "Workload and failure characterization on a large-scale federated testbed," *Intel Research Berkeley, Tech. Rep. IRB-TR-03-040*, 2003.

[20] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[21] *GEANT the pan-european research and education network,*, 2018 (accessed Septemper 10, 2018). [Online]. Available: http://www.geant.net.

[22] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in nfv deployments," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.