Reliability-Aware Service Provisioning in NFV-enabled Enterprise Datacenter Networks

Long Qu
Qatar University
Doha, Qatar
Email: qulong@qu.edu.qa

Chadi Assi
Concordia University
Montreal, Canada
Email: assi@encs.concordia.ca

Khaled Shaban
Qatar University
Doha, Qatar
Email: khaled.shaban@qu.edu.qa

Maurice Khabbaz
Notre-Dame University
Zouk Mosbeh, Lebanon
Email: mkhabbaz@ndu.edu.lb

***Abstract*—Network Function Virtualization (NFV) enables the complete decoupling of Network Functions (NFs) (*e.g.*, firewall, intrusion detection, routing, etc.) from physical middleboxes used to implement service-specific and strictly ordered chains of these NFs. Precisely, NFV allows for dispatching NFs as plain software instances called Virtual Network Functions (VNFs) running on virtual machines hosted by one or more industry standard physical machines. This, however, introduces vulnerabilities (*e.g.*, hard-/soft-ware failures, etc) causing the break down of the entire VNF chain. The functionality of NFV-enabled networks impose higher reliability requirements than traditional networks. This paper encloses an in-depth investigation of a reliability-aware joint VNF placement and flow routing optimization problem. This problem is formulated as a complex Integer Linear Program (ILP). A heuristic is proposed in order to overcome this ILP's complexity. Thorough numerical analysis are conducted to verify and assert the correctness and effectiveness of the proposed heuristic.**

***Index Terms*—NFV, VNF, Reliability, Optimization, Routing.**

I. Introduction

Network Function Virtualization (NFV), complemented by Software-Defined Networking (SDN) technology, presents itself as a revolutionary archetype that leverages virtualization and cloud infrastructure elasticity for the purpose of rendering the network more dynamic, flexible and service-aware [1], [2]. NFV allows consigning Network Functions (NFs) to network operators and service providers in the form of plain software referred to as Virtual Network Functions (VNFs) which can be executed by Virtual Machines (VMs) hosted by one or multiple industry-standard Physical Machines (PMs) located anywhere within and/or at the edge of the network or even at the end-users' premises. Typically, the Service Function Chain (SFC) phenomenon arises from the fact that incoming user traffic is often required to undergo a subset of NFs in a specific order [3]–[5]. An efficient deployment of NFV consists of determining: *i*) the optimal number of required VNF instances and the PMs that will host them (*i.e.,* the VNF placement problem) and *ii*) the optimal end-to-end paths over which traffic flows are routed in order to traverse the required set of placed VNFs (*i.e.,* the routing optimization problem [6]).

The majority of existing work (*e.g.*, [1], [7], [8]) had the objective of minimizing the network's resource consumption for the purpose of maximizing the number of co-existing network services, and hence maximizing the network operator's revenues. The authors of [9] formalized the NF placement and chaining problem and proposed a complex Integer Linear Programming (ILP) model that accounted for additional constraints (*e.g.*, end-to-end latencies, variable processing times and cost-effective resource allocation). However, the above work considered the completely reliable NFV infrastructure (*i.e.*, no network interruptions due to hardware or software failures). This, however, is not realistic. Indeed, the authors of [10] highlight the NFV-related reliability concerns and discuss the types of failures that may arise from both hardware (*e.g.*, restart/shutdown of PMs) and software (*e.g.*, misconfiguration of VMs); hence, causing network service interruptions.

In the context of NFV-enabled networking scenarios, reliable chaining of network services is a problem whose resolution has not yet attained maturity despite the few existing work in the literature (*e.g.* [11], [12]). For instance, the authors of [11] highlighted that the reliability of a Virtual Network (VN) can be derived from the reliability of its components. The work of [12] proposes an online backup selection mechanism. Precisely, first, VNF chains are mapped onto the network's substrate. Then, backup VNFs are selected on the fly if the performed mappings violate the reliability requirements. Furthermore the work of [13] and [14] addressed the problem of reliable Virtual Network Embedding. In particular, the authors of [13] turned their attention to link failure and investigated the reduction of bandwidth utilization through multi-path link embedding. The authors of [14], however, addressed the problem of a single node failure failure-dependent protection in a virtual network embedding framework. Their reported results indicate the absence of any reliability guarantees mainly because only one node serves as a backup node and is shared among multiple principle nodes. A recent study in [15] hints that the dynamic VNF instantiation promotes the protection of the entire network services chain and its failure rate. Nonetheless, failures are still possible and, in the case of a failure, a relatively complex dynamic VNF protection/recovery mechanism must be invoked. This mechanism inevitably introduces nonignorable delays during which the network service remains interrupted; hence, altering the users experience.

Different from the above-surveyed publications, this paper presents a solution that exploits multiple backup nodes for the purpose of provisioning each of the supported network services with adequate reliability guarantees. The problem

978-3-901882-85-2 © 2016 IFIP

of REliability-Aware service CHaining (REACH) in NFV-based networks is investigated herein. A mathematical study is conducted with the objective of formulating and resolving a complex ILP representing REACH's fundamental objectives and constraints. Owing to the formulated ILP's complexity, an algorithmic solution is proposed. Thorough numerical analysis and simulations are conducted to test and validate both of the proposed model and heuristic, and gauge their merit.

II. Network Model

Let S and F denote respectively the set of network services and the overall set of VNFs used by all the services in S. Each service s_i ($i \in [1, |S|]_z$) ($|\cdot|$ and $[a, b]_z$ denote respectively the cardinality of a finite set and the set of integers from a to b) has a reliability requirement Θ^i_{req} ($0 < \Theta^i_{\text{req}} < 1$) and requires its corresponding traffic to traverse an ordered chain composed of a subset of VNFs, F_i ($F_i \subseteq F$). In addition, s_i has a network bandwidth requirement of b_i ($b_i > 0$). Let Σ_i and Δ_i and f_{ij} ($j \in [1, |F_i|]_z$) respectively denote the source, destination (the terms *source* and *destination* are used herein to designate the entry and exit points of an ordered VNF chain) and the j^{th} VNF along the chain of nodes traversed by s_i. Assume the substrate network encompasses a set of PMs N, interconnected by a set of L links. Each PM hosts a set of VNFs. Each VNF requires, for its operation, a VM running on that PM. That is, each VM, is allocated a fraction of its host PM's overall processing capacity C_k ($C_k > 0$) for the purpose of executing the associated VNF. A set of VNFs associated to a PM is represented by an n-tuple $\{f_1, f_2, \cdots, f_n\} \subseteq F$. This means that only VNFs f_1 through f_n can be executed by that PM. Note that any arbitrary VM hosted by an arbitrary PM may execute one VNF and is allocated enough processing capacity c_f ($f \in F$) to do so. In addition, each VNF is prone to failures originating from either hardware or software. Hence, it is associated with a certain reliability measure being its long-term probability of availability. Finally, each pair of VMs, say (n_{k_1}, n_{k_2}), running on different PMs are interconnected by a virtual link m with bandwidth B_m ($B_m > 0$) such that $k_1 = m.head$ and $k_2 = m.tail$.

III. Reliability Awareness

As mentioned earlier, an ordered chain of VNFs is provisioned for each network service for the purpose of enforcing a given policy. A failure of any one of those VNFs along the chain will disrupt the entire chain; hence, causing a violation of the policy to be enforced on the traversing traffic. This problem may have a significant impact, particularly on network services enforcing critical policies (*e.g.*, intrusion detection in an enterprise network). Provisioning policy chains with high service availability requires the selection of highly reliable NFs and constructing chains through these NFs while minimizing the consumption of the network's communication bandwidth. Herein, it is assumed that each PM in the network has a failure rate (*e.g.*, due to software or hardware, power outages, etc.). Thus, VNFs will inherit their respective host PMs' reliability levels. Failures of other network components (*e.g.*, routers, switches, bridges, etc.) are neglected for simplicity. Only PM failures are considered. Consequently, the reliability Θ_i of a service s_i can be computed as the product of the reliabilities of the PMs hosting the VNFs along s_i's chain. According to [16], $\Theta_i = \Pr[\text{all PMs hosting } s_i\text{'s VNFs are available}]$. Thus: $\Theta_i = \prod_{j \in [1, |F_i|]_z} r_j$, $\forall i \in [1, |S|]_z$, where each PM node's reliability can be computed as $r_j = MTBF_j \times (MTBF_j + MTTR_j)^{-1}$. $MTBF_j$ and $MTTR_j$ respectively denote the mean time between failures and the mean time to repair. Per-node failures are independent.

If s_i's achieved reliability, Θ_i, does not meet s_i's required reliability, Θ^i_{req} (*i.e.*, $\Theta_i < \Theta^i_{\text{req}}$), then augmenting s_i's chain with backup VNFs will increase that chain's robustness to possible outages. For instance, upon the occurrence of failures affecting some PMs, backup VNFs hosted by other operational PMs shall serve as substitutes for the stalled VNFs. These backup VNFs are provisioned in a way to maintain the proper order of functions along the chain. That is, a backup VNF should be connected to its protectee's respective predecessor and successor. A sufficient number of redundant VNFs should be provisioned along the chain with the objective of satisfying a network service's reliability requirements. Backup communication bandwidths should also be provisioned along the links connecting the backup VNFs to the service's chain. The impact of augmenting a network service's chain with backup VNFs is additive to the overall availability of that chain. Let F_i be the set of functions along the chain of a service s_i. Each *primary* function $f_j \in F_i$ may have one or more *redundant* functions (of the same type j) provisioned to restore the service (of the chain) when the primary function fails. Let F^T_j denote the set of all functions of type j provisioned for service s_i (*i.e.*, protection domain for VNF j, [16]). F^T_j contains at least one VNF of type j, namely, the primary VNF. Now, the chain's reliability becomes $\Theta_i = \Pr[\text{at least one VNF of each type } j \text{ is available}]$. Hence: $\Theta_i = \prod_{j \in F_i} \left[1 - \prod_{x \in F^T_j} (1 - r_x)\right]$, where r_x is the reliability of a VNF f_x.

IV. System Model

Consider the problem of service chain routing with reliability constraints in NFV-enabled networks. Assume each PM k is hosting a subset of the NFs F. Define $x_{k,t} = 1$ if a function of type t is hosted on PM k. Otherwise $x_{k,t} = 0$. Also, define a binary variable y^k_{ij}($i \in [1, |S|]_z, j \in [1, |F_i|]_z, k \in [1, |N|]_z$) to represent the node assignment of VNFs such that $y^k_{ij} = 1$ if a function j of service i is hosted on PM k. Otherwise, $y^k_{ij} = 0$. To carry out the routing of the service chain, a binary variable z^m_{ij} ($i \in [1, |S|]_z, j \in [1, |F_i|]_z, m \in [1, |L|]_z$) is defined as $z^m_{ij} = 1$ if link m is selected by function j of service i. Otherwise $z^m_{ij} = 0$. REACH has the objective of establishing service chains while minimizing the communication bandwidth usage across the network. Mathematically, this objective is given by:

$$min\{\frac{1}{\sum_{m \in [1, |L|]_z} B_m} \sum_{i \in [1, |S|]_z} \sum_{j \in [1, |F_i|]_z} z^m_{ij} b_i\} \qquad (1)$$

Now, as mentioned earlier, each service chain requires a set of functions F_i, and each function $f_t \in F_i$ acts as a primary and may (or may not) have redundant functions of the same type to restore the service along the chain when the function fails. Therefore, we define NUM_{ij} to be the number of instances of the j^{th} function of service i. Define p_{ij}^k $(i \in [1, |S|]_z), j \in [0, |F_i|+1]_z, k \in [1, |N|_z])$ such that $p_{ij}^k = 1$ if the first instance of s_i's VNF j is hosted on PM k. Otherwise, $p_{ij}^k = 0$. Similarly, define q_{ij}^k $(i \in [1, |S|]_z), j \in [0, |F_i|+1]_z, k \in [1, |N|_z])$ such that $q_{ij}^k = 1$ if the last instance of s_i's VNF j is hosted on PM k. Otherwise, $q_{ij}^k = 0$. Let the indicators 0 and $|F_i|+1$ denote s_i's source, Σ_i, and destination, Δ_i. Therefore, $p_{i,0}^{\Sigma_i} = 1$, $q_{i,0}^{\Sigma_i} = 1$ and $q_{i,|F_i|+1}^{\Delta_i} = 1$. The above definitions of p and q can be used to enforce the fact that only one PM can host the first or last instance of a VNF along a service's chain as follows:

$$\sum_{k \in [1,|N|]_z} p_{ij}^k = 1 \ , \ (\forall i \in [1,|S|]_z, j \in [0, |F_i|+1]_z) \tag{2}$$

$$\sum_{k \in [1,|N|]_z} q_{ij}^k = 1 \ , \ (\forall i \in [1,|S|]_z, j \in [0, |F_i|+1]_z) \tag{3}$$

Constraint (4) below ensures that if the first instance of s_i's VNF j is placed on node k, then there must be an incoming link m to node k, which forwards traffic towards this function.

$$p_{ij}^k \leq \sum_{m.tail=k} z_{ij}^m \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{4}$$

Assume that the node which is selected for the last instance of s_i's VNF j must be hosting a VNF. Then:

$$q_{ij}^k \leq y_{ij}^k \ , \ (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|]_z) \tag{5}$$

Next, it is necessary to enforce the right order for traversing the network functions along the chain. For this reason. the chain provisioning procedure is decomposed into two sub-problems. The first sub-problem consists of handling the routing constraints within the instances/copies of the same VNF in a network service chain (namely VNF routing constraints). Based on flow balance criteria, the incoming flow must equal the outgoing flow at VM or PM nodes for each network service. Then, the VNF routing is formulated as follows:

$$\sum_{m.head=k} z_{ij}^m - \sum_{m'.tail=k} z_{ij}^{m'}(1 - q_{ij}^k) = q_{ij-1}^k \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{6}$$

$$\sum_{m.tail=k} z_{ij}^m \leq 1 \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [0, |F_i|+1]_z) \tag{7}$$

$$\sum_{m.head=k} z_{ij}^m \leq 1 \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [0, |F_i|+1]_z) \tag{8}$$

$$z_{ij}^m z_{ij}^{m'} = 0 \qquad (\forall m, m' \in [1,|L|]_z, m.head = m'.tail, m.tail = m'.head, i \in [1,|S|]_z, j \in [0, |F_i|+1]_z) \tag{9}$$

where (6) makes sure that, for one node, the flow balance must be satisfied if the current node is not implementing or hosting the last instance/copy of VNFs j and $j-1$ (e.g., q_{ij}^k = 0 and q_{ij-1}^k = 0). Equations (7), (8) and (9) make sure that the assigned routes are loop free.

Next, the routing constraints between different VNFs of one network service (namely, the service routing constraints) are formulated as the second sub-problem. For this purpose, an auxiliary binary variable w_{ij}^m is defined such that $w_{ij}^m = 1$ if link m connectes s_i's VNFs j and $j-1$. Otherwise, $w_{ij}^m = 0$. Then, the service routing constraints can be written as:

$$w_{ij}^m = q_{ij-1}^{m.head} p_{ij}^{m.tail} \qquad (\forall m \in [1,|L|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{10}$$

$$w_{ij}^m \leq z_{ij}^m \qquad (\forall m \in [1,|L|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{11}$$

where (10) ensures $w_{ij}^m = 1$ if and only if both $q_{ij-1}^{m.head} =$ 1 and $p_{ij}^{m.tail}$. Equation (11) forces a link m to be selected ($z_{ij}^m = 1$) if the indicator $w_{ij}^m = 1$.

To make sure the processes of VNF instances are independent in a network service (*i.e.*, to simplify the computation of the reliability), VNF instances belonging to the same network service are assumed to be implemented on different VMs or PMs. This is formulated as:

$$q_{ij-1}^k p_{ij}^k = 0 \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{12}$$

Note that (6) through (12) are non-linear constraints, due to the product of binary variables. An auxiliary binary variable $zq_{ij}^m = z_{ij}^m q_{ij}^k$ $(\forall m \in [1,|L|]_z, k = m.tail, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z)$ is used to convert (6) into a linear equation:

$$\sum_{m.head=k} z_{ij}^m - \sum_{m'.tail=k} z_{ij}^{m'} + \sum_{m'.tail=k} zq_{ij}^{m'} = q_{ij-1}^k \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{13}$$

Now, the relationship between zq_{ij}^m and z_{ij}^m, q_{ij}^k becomes:

$$z_{ij}^m \geq zq_{ij}^m \tag{14}$$

$$q_{ij}^k \geq zq_{ij}^m \tag{15}$$

$$zq_{ij}^m \geq z_{ij}^m + q_{ij}^k - 1 \tag{16}$$

The similar linearization of (9) through (12) is omitted.

Next, the reliability constraints are introduced as follows:

$$\sum_{k \in [1,|N|]_z} y_{ij}^k \geq NUM_{ij} (\forall i \in [1,|S|]_z, j \in [1, |F_i|]_z) \tag{17}$$

$$y_{ij}^k \leq \sum_{m.head=k} z_{ij}^m \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|+1]_z) \tag{18}$$

$$y_{ij}^k \leq x_{k,t} \qquad (\forall k \in [1,|N|]_z, i \in [1,|S|]_z, j \in [1, |F_i|]_z) \tag{19}$$

with t being the type of VNF f_{ij}

$$\sum_{i \in [1,|S|]_z} \sum_{j \in [1,|F_i|]_z} y_{ij}^k c_{f_{ij}} \leq C_k (\forall k \in [1,|N|]_z) \tag{20}$$

$$\sum_{i\in[1,|S|]_z}\sum_{j\in[1,|F_i|]_z} z_{ij}^m b_i \leq B_m(\forall m\in[1,|L|]_z) \qquad (21)$$

To achieve the reliability requirement of a network service, (17) ensures the provisioning of enough instances (NUM_{ij}) of a particular VNF for a particular service (An iterative method is discussed hereafter to decide on the value of NUM_{ij}). Constraints (18) and (19) indicate the relationship between the routing variables and the VNF placement. Finally, (20) and (21) enforce the limit on the nodes' processing resources and link bandwidth capacity.

V. REACH: Solution Methodology

A. ILP-based Solution with Redundant VNFs:

Given a network service's reliability requirement (Θ_{req}^i), REACH attempts to provision that service with a chain of VNFs that meets this requirement with minimal consumption of link bandwidth and processing resources. The steps for generating this solution are summarized in Algorithm 1. First, assume that only primary VNFs (*i.e.*, $NUM_{ij} = 1$) are provisioned along the chain. If the provisioned chain meets the network service's requested reliability, then stop. The obtained solution consumes the least bandwidth and processing resources. Otherwise, if no solution incorporating only primary VNFs can be found to satisfy the requirements, then redundant VNFs must be adequately added along the chain. Precisely, select the least reliable VNF along the chain and enhance its reliability through the instantiation of a backup copy of it (*i.e.*, $NUM_{ij} = NUM_{ij} + 1$ in line 13 of Algorithm 1). Here NUM_{ij} is a parameter owing to the incremental update of its value throughout each iteration. Let r_{ij} $(i\in[1,|S|]_z, j\in[1,|F_i|]_z)$ denote the reliability of VNF j along the chain corresponding to network service i. The relationship can be written as:

$$r_{ij} = 1 - \prod_{k\in[1,|N|]_z}(1 - y_{ij}^k\theta_k) \qquad (22)$$

The above process is repeated as long as the reliability requirement of the provisioned chain remains unsatisfied. Otherwise, it stops.

Algorithm 1: ILP with redundant VNFs

```
1  Initialization:
2    Substrate Network ($N$) and Network Services ($S$);
3    $NUM_{ij} = 1(\forall i\in[1,|S|]_z, j\in[1,|F_i|]_z)$;
4    $isFinished = 1$;
5  while $true$ do
6      Solve the optimal problem:
7      Objective: (1)
8      Constraints: (2) - (21) (Linearization)
9      Calculate the VNFs reliability $r_{ij}$;
10     for $i = 1 : |S|$ do
11         if $\prod_{j\in(1,|F_i|_z)} r_{ij} < \Theta_{req}^i$ then
12             Select the VNF $j$ with the weakest reliability;
13             $NUM_{ij} = NUM_{ij} + 1$;
14             $isFinished = 0$;
15         end
16     end
17     if $isFinished = 1$ then
18         break;
19     end
20     $isFinished = 1$;
21 end
```

B. Greedy Shortest Path Solution with Redundant VNFs:

In this section, a light weight greedy shortest path approach (*i.e.*, Algorithm 2) is presented as a substitute for solving the heavy weight ILP model used to chain functions in each of Algorithm 1's iterations (*i.e.*, lines 6 through 8). In Algorithm 1, the model is used to generate a feasible routing with the least resource cost for network services. That is to select one feasible node (*i.e.*, a node having enough computational resources and, hence, is capable of executing the VNF) as a host for each VNF instance and establish a route between these VNFs with a reasonable low bandwidth consumption. At this point, two variables $curr$ and $next$ are introduced to describe the order of VNF processing. Precisely, the variable $curr$ indicates the VNF instance that is currently being handled by the algorithm performing node assignment whereas the variable $next$ indicates the VNF instance that is going to be assigned next. This present algorithm starts from the source of each network service ns (*i.e.*, $ns.source$) and terminates at that network service's destination node (*i.e.*, $ns.dest$). First, the algorithm initializes $curr = ns.source$. Then, shortest path routing (*e.g.*, Dijkstra's algorithm) is invoked for the purpose of finding the node assignment for $next$ = $(f_{ij}^b)_n$ (namely, the VNF instances/backups of f_{ij}). Following this comes the calculation of the bandwidth costs from $curr$ to the rest of feasible PMs (*i.e.*, machines with appropriate VNFs) except the one which has been occupied by $curr$ or is currently not available due to the lack of VNF capacity or processing resources. The node incurring the least bandwidth consumption is selected as the assignment for $next$. Afterwards, the next VNF is visited (*i.e.*, $curr = next$). This process continues until all VNF instances have been assigned to PMs.

VI. Numerical Results

This section is dedicated for the evaluation of REACH's performance. An NFV-enabled Reliability Aware Routing and resource optimization scheme (RAR-NFV, Algorithm 1) is considered together with a Greedy shortest path method (Greedy-NFV, Algorithm 2). The CPLEX solver is used to solve the ILP model pertaining RAR-NFV. All our simulations are implemented and solved on a machine equipped with an Intel 2.6 GHz processor and 8 GB RAM. Throughout this performance evaluation framework, two network instances are considered, namely: *i*) a smaller network composed of 16 PMs hosting 4 network services and *ii*) a larger network composed of 40 PMs hosting 10 to 40 network services. Without loss of generality, each network service requires a certain number of

Algorithm 2: Greedy Shortest Path With Redundant VNFs

```
1  Given:
2    Substrate Network (N), Network Services (S) and
3    $NUM_{ij}$ from Algorithm 1
4  for $i = 1 : |S|$ do
5      Initialize $curr = i.source$;
6      for $j = 1 : |F_i|$ do
7          for $n = 1 : NUM_{ij}$ do
8              $next = (f_{ij}^b)_n$;
9              Node bandwidth costs = Dijkstra($curr$);
10             Select the feasible node with least bandwidth
               cost as the host for $next$;
11             Update substrate network N by deducting the
               bandwidth and CPU resource consumption;
12             $curr = next$;
13             if $j = |F_i|$ && $n = NUM_{ij}$ then
14                 Dijkstra ($curr$);
15                 Select shortest path from $curr$ to $i.dest$;
16                 Update bandwidth and CPU
                   consumptions;
17             end
18         end
19     end
20 end
```

VNFs (fixed to 4 throughout our simulations). For the purpose of promoting the problem's tractability, each PM is assumed to have a capacity enabling it to host 2 to 3 VNFs and each VNF can at least be processed on one PM. The reliabilities of PM nodes are randomly generated between 0.9 and 0.96. In the 16-node network, the CPU consumption for each VNF is set to 2 (units). The nodal computational capacities and the link bandwidths are randomly drawn from the range of 10 to 20 (units). The four network services are: **NS1:** $f_3 \rightarrow f_2 \rightarrow f_4 \rightarrow f_1$ (Σ_1: 15; Δ_1: 2); **NS2:** $f_2 \rightarrow f_1 \rightarrow f_4 \rightarrow f_3$ (Σ_2: 5; Δ_2: 16); **NS3:** $f_1 \rightarrow f_3 \rightarrow f_2 \rightarrow f_4$ (Σ_3: 5; Δ_3: 14); **NS4:** $f_4 \rightarrow f_2 \rightarrow f_3 \rightarrow f_1$ (Σ_4: 8; Δ_4: 13).

For a fair comparison, the reliability and bandwidth requirements of the network services are respectively set to 0.99 and 2. The obtained results are tabulated in Table 1. It is clear that both RAR-NFV and Greedy-NFV achieve the reliability requirement of the network services (*i.e.*, 0.99), with both methods achieving similar overall bandwidth consumption. Not surprisingly, Greedy-NFV shows much better scalability than RAR-NFV. This is especially true since Greedy-NFV is able to find solutions within only 0.32 seconds versus 14028 seconds (even for this small network instance) for RAR-NFV. This is, indeed, due to the fact that RAR-NFV resorts to solving the ILP model for routing the chains (with constant NUM_{ij}) at each iteration of the algorithm. For this reason, in the remaining of this section, Greedy-NFV is used for all purposes of performance evaluation for larger networks.

Next, a 40-node networking scenario is considered. In this context, the provisioning of network services with different reliability requirements is studied. The reliability requirement of network services is varied in the range of 0.98 to 0.992. The network is assumed to host a total of 20 random services. Each PM is equipped with a certain number of VNFs (*i.e.*, 2 to 3 VNFs) and each VNF requires 2 to 4 units of CPU resources. The link bandwidth and node computational capacity are randomly drawn from a range of 20 to 40 units respectively. Each network service requires 2 bandwidth units.

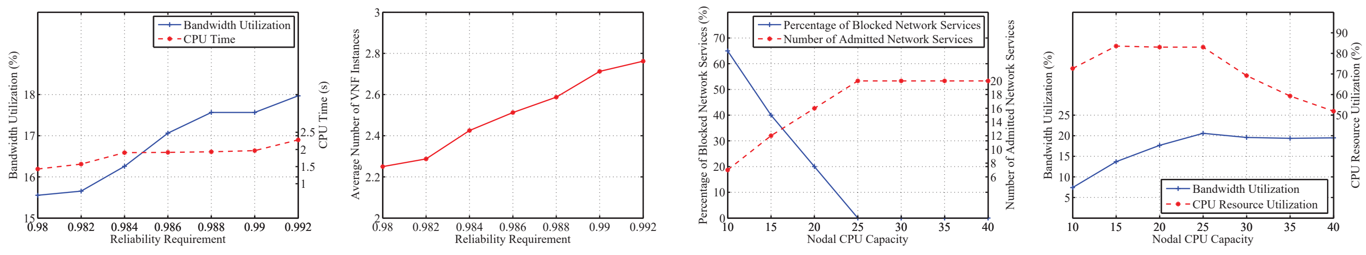
Figure 1(a) plots the network-wide bandwidth utilization as a function the network services' reliability requirements. The average number of needed VNF instances is plotted as a function of the different reliability requirements in Figure 1(b). As it was shown earlier, a higher reliability requirement can be satisfied by instantiating redundant VNFs along the service chains. This will require additional links to connect the redundant functions to the chain and, hence, bandwidth must be provisioned along those links. Therefore, as illustrated in Figure 1(a), as the services' reliability requirements increases, more bandwidth throughout the network needs to be provisioned in order to allow each redundant VNF to communicate with its immediate upstream and downstream VNFs along the established chain. Figure 1(a) shows the CPU runtime of Greedy-NFV. Clearly, this algorithm exhibits fast runtimes, which promotes its practical utility in provisioning reliable chains in NFV-based networks.

Now, using the same above network setting (the reliability requirement is fixed as 0.996), in a 40-node network with 20 network services, the impact of varying the nodal CPU capacity (*i.e.*, 10 to 40 units) is investigated where nodes with 10 units and 40 units CPU capacities can be respectively interpreted as implementing NFVs over inexpensive and expensive PMs. Intuitively, the implementation of VNF instances costs nodal CPU resources. If there is not enough CPU resources, the network services will be dropped. Figure 1(c) shows the network services' loss ratio. Figure 1(d) shows the nodal CPU and link bandwidth utilizations. Observe that the loss ratio goes as high as 65% over inexpensive PMs. With the increase of nodal CPU capacity, more and more network services can be admitted and the loss ratio drops down to zero. In other words, in order to admit 20 network services without any loss, PM nodes must have at least 25 units of CPU capacity. The results of CPU utilization follow the same trend. When nodal CPU capacity less than 25 units, the CPU utilization goes up to 83%. This indicates that the lack of CPU resources in the network causes the loss of network services. However, a PM node with very large CPU capacity (*e.g.*, 40 units) is not necessary and will incur relatively high additional costs. Also notice that the bandwidth consumption corresponding to a nodal capacity of 30 is less than the one corresponding to a nodal capacity of 25 irrespective of the fact that, in both cases, the loss ratio is zero. This is because the nodal capacity has an effect on the routing decision. Hence, in order to achieve a reliable NFV implementation, it is important to select a platform with reasonable computational resources.

Next, the impact of varying the network load (*i.e.*, the volume of network services to be provisioned) is investigated

TABLE I
ROUTING RESULTS FOR A 16-NODE NETWORK

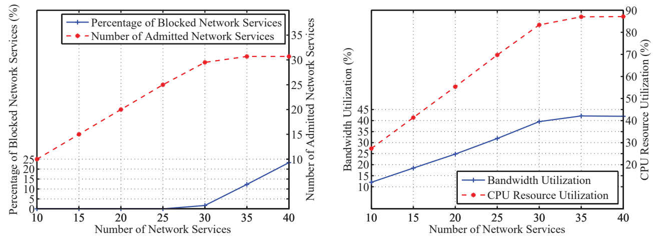
Algorithm	NS	Routing and VNFs assignment (VM (VNFs))	Reliability	Bandwidth Utilization	CPU time (s)
RAR-NFV	1	$15 \rightarrow 14(f_3) \rightarrow 13(f_3) \rightarrow 9(f_2) \rightarrow 5(f_2) \rightarrow 1(f_2) \rightarrow 2(f_4) \rightarrow 6(f_4) \rightarrow 10 \rightarrow 11(f_1) \rightarrow 7(f_1) \rightarrow 3(f_1) \rightarrow 2$	0.9936	15.47%	14028
	2	$5 \rightarrow 6 \rightarrow 7(f_2) \rightarrow 8(f_2) \rightarrow 4(f_2) \rightarrow 3(f_1) \rightarrow 2(f_1) \rightarrow 6(f_4) \rightarrow 10(f_4) \rightarrow 9(f_3) \rightarrow 13(f_3) \rightarrow 14(f_3) \rightarrow 15 \rightarrow 16$	0.9907		
	3	$5 \rightarrow 1 \rightarrow 2(f_1) \rightarrow 6(f_1) \rightarrow 10(f_3) \rightarrow 14(f_3) \rightarrow 13(f_3) \rightarrow 9(f_2) \rightarrow 5(f_2) \rightarrow 6 \rightarrow 7(f_4) \rightarrow 11(f_4) \rightarrow 15(f_4) \rightarrow 14$	0.9924		
	4	$8 \rightarrow 7(f_4) \rightarrow 6(f_4) \rightarrow 10(f_4) \rightarrow 9(f_2) \rightarrow 5(f_2) \rightarrow 1(f_2) \rightarrow 2 \rightarrow 3(f_3) \rightarrow 4(f_3) \rightarrow 8(f_3) \rightarrow 12(f_1) \rightarrow 11(f_1) \rightarrow 15 \rightarrow 14(f_1) \rightarrow 13$	0.9949		
Greedy-NFV	1	$15 \rightarrow 14(f_3) \rightarrow 13(f_3) \rightarrow 9(f_2) \rightarrow 5(f_2) \rightarrow 1(f_2) \rightarrow 2 \rightarrow 6(f_4) \rightarrow 7(f_4) \rightarrow 11(f_1) \rightarrow 12(f_1) \rightarrow 8 \rightarrow 4 \rightarrow 3(f_1) \rightarrow 2$	0.9932	15.47%	0.32
	2	$5 \rightarrow 9(f_2) \rightarrow 5(f_2) \rightarrow 1(f_2) \rightarrow 2(f_1) \rightarrow 6(f_1) \rightarrow 7(f_4) \rightarrow 11(f_4) \rightarrow 15(f_4) \rightarrow 14(f_3) \rightarrow 13(f_3) \rightarrow 14 \rightarrow 15 \rightarrow 16$	0.9937		
	3	$5 \rightarrow 6(f_1) \rightarrow 2(f_1) \rightarrow 3(f_3) \rightarrow 4(f_3) \rightarrow 8(f_3) \rightarrow 7(f_2) \rightarrow 11 \rightarrow 15(f_2) \rightarrow 11(f_4) \rightarrow 10(f_4) \rightarrow 9 \rightarrow 13(f_4) \rightarrow 14$	0.9939		
	4	$8 \rightarrow 7(f_4) \rightarrow 11(f_4) \rightarrow 15(f_2) \rightarrow 11 \rightarrow 12(f_2) \rightarrow 8(f_3) \rightarrow 4(f_3) \rightarrow 3(f_3) \rightarrow 2(f_1) \rightarrow 6(f_1) \rightarrow 5 \rightarrow 9 \rightarrow 13$	0.9903		



(a) Bandwidth utilization. (b) Mean number of VNF instances. (c) Loss ratio. (d) Bandwidth and CPU utilization.

Fig. 1. Results pertaining to a 40-node network with 20 network service.

in the context of a networking scenario with finite resources (*i.e.*, CPU capacity and link bandwidth). The reliability requirement of the services is fixed to 0.99 and the capacity per link is fixed to 20 (units). The number of services is varied from 10 (*i.e.*, light load) to 40 (*i.e.*, high load), and these flows are randomly generated. For all purposes of comparison fairness, ten algorithmic runs are executed and their results are averaged out. Given the limit on the network resources, it is expected that the number of blocked flows (*i.e.*, flows that are not admitted to the network) increases as a function of the load. This is especially true since, beyond a certain load threshold, the network will start blocking service requests, as it is not able to satisfy their requirements due to the lack of resources. This is clearly illustrated in Figure 2(a). Finally, the network-wide bandwidth and CPU utilizations are evaluated and plotted in Figure 2(b) as a function of the network load. The figure shows that the CPU utilization increases to 83.36% (*i.e.*, almost exhausting the available processing resources per PM) while maintaining a loss ratio as low as 1.67% (*i.e.*, the number of network services ranges from 10 to 30). However, observe here that the number of admitted network services is very close to its maximum (*i.e.*, 40). Hence, Greedy-NFV is exploiting almost the full CPU capacity as it admits almost the maximum number of network services.



(a) Loss ratio. (b) Bandwidth and CPU utilization.

Fig. 2. Results pertaining to a 40-node network with service reliability requirement of 0.99).

VII. CONCLUSION

This paper presents a novel REliability-Aware service CHaining (REACH) framework for NFV-enabled enterprise networks. A Reliability-Aware Routing and resource allocation (RAR-NFV) scheme implementing the ILP's solution obtained using the CPLEX solver is developed. To overcome RAR-NFV's complexity, a Greedy shortest-path-based (Greedy-NFV) heuristic is proposed. Simulations are conducted using the Greedy-NFV heuristic to gauge REACH's merit and evaluate its performance in terms of CPU and network-wide bandwidth capacity utilization.

ACKNOWLEDGMENT

This work was made possible by the NPRP 5-137-2-045 grant from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] M. F. Bari, (*et. al.*), ”On Orchestrating Virtual Network Functions in NFV,” arXiv: 1503.06377v2 [cs.NI], Mar. 2015.

[2] R. Mijumbi, (*et. al.*), ”Network Function Virtualization: State-of-the-art and Research Challenges,” *IEEE Comm. Surv. & Tut.*, 18:1, 2015.

[3] W. Liu, (*et. al.*), ”Service Function Chaining (SFC) General Use Cases,” *IETF - Draft*, URL: http://tools.ietf.org/html/draft-liu-sfc-use-cases-08, 2014.

[4] W. Haeffner, (*et. al.*), ”Service Function Chaining Use Cases in Mobile Networks,” *Service Function Chaining - Internet Draft* [Available Online] URL: http://tools.ietf.org/html/draft-haeffner-sfc-use-case-mobility-00, Jan, 2014.

[5] S. Kumar, (*et. al.*), ”Service Function Chaining Use Cases In Data Centers,” *IETF - Draft* URL: http://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-03, 2015.

[6] B. Addis, (*et. al.*), ”Virtual Network Functions Placement and Routing Optimization,” *Proc. IEEE CLOUDNET*, ON, Canada, 2015.

[7] S. Mehraghdam, (*et. al.*), ”Specifying and Placing Chains of Virtual Network Functions,” *Proc. IEEE CLOUDNET*, Luxembourg, Luxembourg, Oct. 2014.

[8] H. Moens, (*et. al.*), ”VNF-P: A Model For Efficient Placement of Virtualized Network Functions,” *Proc. of IEEE CNSM*, Rio De Janeiro, Brazil, Nov. 2014

[9] M. C. Luizelli, (*et. al.*), ”Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions,” *Proc. IFIP/IEEE IM*, Ottawa, Ontario, Canada, May 2015.

[10] D. Cotroneo, (*et. al.*), ”Network Function Virtualization: Challenges And Directions for Reliability Assurance,” *Proc. IEEE ISSREW*, Italy, 2014.

[11] R. Guerzoni, (*et. al.*), ”Modeling Reliability Requirements In Coordinated Node And Link Mapping” , *Proc. IEEE SRDS*, Japan, 2014.

[12] J. Fan, (*et. al.*), ”GREP: Guaranteeing Reliability With Enhanced Protection in NFV,” *Proc. ACM SIGCOMM Workshop on HotMiddlebox*, London, U.K., 2015.

[13] M.M.A Khan, (*et. al.*), ”SiMPLE: Survivability in Multi-path Link Embedding” , *Proc. IEEE CNSM*, Barcelona, Spain, 2015.

[14] B. Guo, (*et. al.*), ”Survivable Virtual Network Design and Embedding to Survive a Facility Node Failure,” *Journal of Lightwave Technology.*, 32:3, 2014.

[15] A. Gember-Jacobson, (*et. al.*), ”OpenNF: Enabling innovation in network function control”, *ACM SIGCOMM Computer Communication Review*, 44(4), pp.163-174, 2015.

[16] S. Ayoubi, (*et. al.*), ”RAS: Reliable Auto-Scaling of Virtual Machines In Multi-Tenant Cloud Networks” , *Proc. IEEE CLOUDNET*, ON, Canada, 2015