Hardware-accelerator aware VNF-chain recovery

Gourav Prateek Sharma, Wouter Tavernier,
Didier Colle, Mario Pickavet
IDLab, Department of Information Technology
Ghent University - IMEC,
Email: {gouravprateek.sharma, wouter.tavernier, didier.colle, mario.pickavet}@ugent.be

***Abstract*—Hardware-accelerators in Network Function Virtualization (NFV) environments have aided telecommunications companies (telcos) to reduce their expenditures by offloading compute-intensive VNFs to hardware-accelerators. To fully utilize the benefits of hardware-accelerators, VNF-chain recovery models need to be adapted. In this paper, we present an ILP model for optimizing prioritized recovery of VNF-chains in heterogeneous NFV environments following node failures. We also propose an accelerator-aware heuristic for solving prioritized VNF-chain recovery problems of large-size in a reasonable time. Evaluation results show that the performance of heuristic matches with that of ILP in regard to restoration of high and medium priority VNF-chains and a small penalty occurs only for low-priority VNF-chains.**

***Index Terms*—NFV, hardware accelerator, Recovery, SFC, FPGA, placement algorithm, allocation**

I. Introduction

In the past few years, exponential growth of the Internet data traffic has taken place due to the explosion in the number of total number of the connected-users and network services. This change has compelled telecommunications companies (telcos) to look for alternative network-architecture solutions which are more economical, manageable and scalable. Network Function Virtualization (NFV) proposes to transform the manner in which network-services are currently created and managed by leveraging Information Technology (IT) virtualization technologies for network-services. With NFV, services which were earlier implemented using proprietary hardware appliances (middleboxes) can now be created using virtual network functions (VNFs) or Cloud-native Network Functions (CNFs). As a result, telcos can significantly reduce their capital expenditures (CAPEX) by running VNFs (or CNFs) on commercial-off-the-shelf (COTS) servers (e.g. x86 or ARM) instead of purchasing costly middleboxes.

In the recent past, hardware-accelerators like Graphics Processing Units (GPUs), smartNICs and Field Programmable Gate Arrays (FPGAs) are also being included in the NFV infrastructure (NFVi) to address the performance issues with VNFs [1]. Hardware-accelerators can offload the compute-intensive tasks from a VNF running on the CPU resulting in a speedup of execution. This packet-processing offload also frees-up CPU cores which can be used to run other VNFs resulting in better server-consolidation.

In order to truly exploit the benefits of NFV, it is essential to allocate NFVi resources to VNFs as efficiently as possible. State-of-the-art (SotA) VNF placement models and strategies only consider usual NFVi infrastructure resources like compute, storage and network while making their allocation decisions. This can lead to an inefficient allocation of NFVi resources in heterogeneous NFVi which contain hardware-accelerators along with the usual NFVi resources.

Reliability of network-services can be enhanced by utilizing hardware-accelerators to satisfy the diverse Quality of Service (QoS) requirements of service-chains. Therefore, QoS is not only impacted by re-allocation of usual NFVi resources (compute, storage and network) after a failure but also by the hardware-accelerator re-allocation. In the case of server-node failures, the VNF-chain recovery processes should also take into account the presence of hardware-accelerator resources in NFVi along with usual NFVi resources. Moreover, VNF remapping and accelerator-allocation should be able to prioritize traffic according to the QoS requirements. To the best of our knowledge, no work has addressed this problem in the past. To this end, we make the following contributions in this paper:

1) Modeling of prioritized VNF-chain recovery problem in heterogeneous NFV environments using Integer Linear Programming (ILP) approach.
2) Designing greedy-based heuristic algorithm to solve the above problem.
3) Evaluations of ILP and heuristic algorithm.

The rest of the paper is organized as follows. The related work for this paper is discussed in the next section. The problem of accelerator-aware VNF-chain recovery problem is formulated using ILP in section III. Afterwards, a heuristic to solve VNF-chain recovery problem is presented in section IV. Section V describes the evaluation of ILP and heuristics. Finally, the conclusion of the paper is presented in section VI.

II. Related Works

With the growing interest around NFV among industry and academia, various studies have been carried out related to modelling of resource allocation for VNFs. Authors in [2] defined the VNF placement and routing optimization problem and devised a mixed-ILP (MILP) formulation for it. Bari et al. also modeled the VNF orchestration problem (VNF-OP) using ILP approach [3]. They proposed a dynamic programming

approach to solve VNF-OP instances with larger sizes. Authors in [4] have proposed a Hardware Acceleration Resource Allocation Mechanism (UHAD) for simplifying the integration of hardware-accelerators in NFVi.
In [5], the authors studied the problem of allocating backup resources for VNFs and virtual-links such that reliability constraints of service-chains are adhered. A scalable heuristic algorithm, aimed at sharing backup resources in order to reduce overhead costs due to backups, was also proposed. A. Tomassilli et al. investigated two different (dedicated and shared) protection mechanisms for reliable chaining of VNFs in case of a single-link failure [6]. Evaluation of ILP models show that the dedicated protection scheme requires more 40% bandwidth and 60% processing resources as compared to the shared protection scheme. A study regarding service reliability using VNF migration and replications was conducted in [7]. The authors recommended jointly performing VNF migration and replication in order to efficiently utilize server and link resources. An algorithm is also proposed for optimizing the provisioning of hardware-accelerators in NFVi nodes. We also proposed a scheme for dynamic allocation of hardware-accelerators to VNFs in NFV environments by the use of specific service managers (SSMs) [8].
Although numerous resource allocation models have been proposed aiming at optimizing various parameters like cost, performance, load-balancing, etc, there has been a little focus on prioritized VNF-chain placement models for heterogeneous NFV environments. In [9], the authors addressed the problem of joint scaling, placement and routing for heterogeneous services. They presented a MILP approach for single-step exact solutions along with a heuristic algorithm for fast and near-optimum solutions. We formulated the accelerator-aware VNF placement problem in the form of an ILP and also proposed a scalable algorithm based on best-fit heuristic [10].
In summary, the challenge of prioritized VNF-chain recovery in a heterogeneous environment in case of server-node failures still remains unaddressed.

III. Problem Formulation

The description of various parameters and decision variables involved in the ILP formulation is given in Table I.
The objective of this ILP formulation is to maximize the total amount of traffic restored after a failure subject to the resource capacity. In addition to that, restoration is prioritized based on traffic class of VNF-chains. In the objective function (1), x_s is the decision variable indicating if the VNF-chain s is restored after the failure. The product $\mu_s x_s$ denotes the amount of traffic restored after the recovery of VNF-chain s. The scaling factor μ_s in the objective function is the cost-gain for restoring a service-chain $s \in S$. The prioritization of VNF-chains is performed by calculating μ_s of VNF-chains based on their respective traffic-class as explained below.
Depending on the traffic-class of VNF-chains, all VNF-chains are categorized into three sets, namely– high-priority ($S_I \subseteq S$), medium-priority ($S_{II} \subseteq S$) and the low-priority ($S_{III} \subseteq S$) VNF-chains. After any node-failure in NFVi, the preference for restoration is given to VNF-chains $s \in S$ in the following order: first for $s \in S_I$, then for $s \in S_{II}$, and at last for $s \in S_{III}$. The requirement for prioritizing the VNF-chain restoration can be implemented by adding soft-constraints to the ILP formulation. This is done through the assignment of restoration gain μ_s for each VNF-chain s as indicated in (2). Here T_{II} and T_{III} can be evaluated using (3). By scaling the throughput values of traffic ($x_s t_s$) for each VNF-chain request with a restoration-gain value μ_s, we have avoided the requirement for adding hard-constraints with regards to prioritized restoration of traffic.

$$obj : \max \sum_{s \in S} t_s \mu_s x_s \qquad (1)$$

TABLE I
Description of parameters and decision variables

Input parameters	
Notation	Description
N	Set of all non-failed computational nodes within NFVi.
$\mathcal{R}_{cpu}(n)$	Maximum CPU resources (cores) available on $n \in N$.
$\mathcal{R}_{acc}(n)$	Maximum accelerator fabric resources (logic elements) available on $n \in N$.
$\mathcal{R}_{bus}(n)$	Maximum bandwidth (Gbps) of the PCIe bus of node $n \in N$.
A	Set of all available accelerator types.
$r(a)$	Resource requirement (logic elements) of the accelerator type $a \in A$.
S	Set of all service requests, including top (S_I), medium (S_{II}) and low (S_{III}) priority services.
F^s	Set of all VNFs corresponding to the service-chain request $s \in S$.
t_s	Throughput requirement (Gbps) of the service request $s \in S$.
$cpu_0(f^s)$	CPU requirement (cores) of VNF f of the service request $s \in S$.
$cpu_r(f^s)$	CPU reduction (cores) for VNF f of the service request $s \in S$.
$atype(f^s)$	Type of accelerator needed for acceleration of VNF f of the service request $s \in S$.

Decision variables	
Notation	Description
x_s	Binary decision variable indicates if the service $s \in S$ is restored after the failure.
α^n_{fs}	Binary variable indicates if VNF f of service request s is placed on n after the failure.
β^n_{fs}	Binary variable indicates if VNF f of service request s is accelerated on n after the failure.
δ^n_a	Binary variable indicates if accelerator of type a is instantiated on the node n after the failure.

$$\begin{gathered}\mu_{s_{III}} = 1, \quad \mu_{s_{II}} t_{s_{II}} = T_{III}, \quad \mu_{s_I} t_{s_I} = T_{II} + T_{III} \\ \forall s_{III} \in S_{III}, \quad s_{II} \in S_{II}, \quad s_I \in S_I\end{gathered} \qquad (2)$$

$$T_{III} = \sum_{s \in S_{III}} \mu_s t_s, \quad T_{II} = \sum_{s \in S_{II}} \mu_s t_s \qquad (3)$$

A. Node constraints

For each server-node $n \in N$, total number of CPU cores utilized by all VNFs placed on n is constrained by the number of cores available on it as indicated in (4). The constraint on resources available on the hardware-accelerator fabric for the instantiation of accelerators is shown in (5). Constraint in (6) ensures PCIe bandwidth required for communication between VNFs and accelerators does not exceed the available bandwidth.

$$\sum_{s\in S, f^s\in F^s} \alpha^n_{f^s} cpu_0(f^s) - \beta^n_{f^s} cpu_r(f^s) \leq \mathcal{R}_{cpu}(n) \quad \forall n \in N \tag{4}$$

$$\sum_{a\in A} r(a)\delta^n_a \leq \mathcal{R}_{acc}(n) \quad \forall n \in N \tag{5}$$

$$\sum_{s\in S, f^s\in F^s} 2t_s\beta^n_{f^s} \leq \mathcal{R}_{bus}(n) \quad \forall n \in N \tag{6}$$

B. Acceleration constraints

The constraint in (7) is a consequence of the fact that a VNF f^s can only be allocated an accelerator on n if the f^s is placed on n. Also, binary decision variable δ^n_a is assigned a value of 1 if atleast one VNF is allocated accelerator a on n as indicated in (8). This constraint can be linearized by the re-formulation shown in (9a-9b). If no VNF is allocated accelerator a on n, (9a) forces δ^n_a to be equal to zero. Conversely, if atleast one VNF is assigned accelerator a on n, LHS of (9b) is ≥ 1 resulting in δ^n_a to take a value 1. The factor M_1 in the RHS of (9b) is a constant greater than the total number of VNFs in all VNF-chains, i.e., $M_1 = \sum_{\substack{\forall s\in S\\ \forall f^s\in F^s}} 1$.

$$\beta^n_{f^s} \leq \alpha^n_{f^s} \quad \forall n \in N, \forall s \in S, \forall f^s \in F^s \tag{7}$$

$$\delta^n_a = \begin{cases} 1, & \text{if} \sum_{\substack{\forall s\in S, \forall f^s\in F^s,\\ a=atype(f^s)}} \beta^n_{f^s} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N, \forall a \in A \tag{8}$$

$$\delta^n_a \leq \sum_{\substack{\forall s\in S, \forall f^s\in F^s\\ a=atype(f^s)}} \beta^n_{f^s} \quad \forall a \in A,\ \forall n \in N \tag{9a}$$

$$\sum_{\substack{\forall s\in S, \forall f^s\in F^s\\ a=atype(f^s)}} \beta^n_{f^s} \leq M_1\delta^n_a \quad \forall a \in A,\ \forall n \in N \tag{9b}$$

C. Other Constraints

A VNF-chain is said to be restored if all the VNFs constituting the VNF are placed on computational nodes. This requirement is expressed by a set of constraints given in (10). This constraint can be linearized using the method discussed before for linearizing the constraint in (8). Constraints in (11) force binary variables $x_s, \alpha^n_{f^s}, \beta^n_{f^s}, \delta^n_a$ to only take binary values (0 or 1).

$$x_s = \begin{cases} 1, & \text{if} \sum_{\forall n\in N} \alpha^n_{f^s} = 1, \quad \forall f^s \in \mathcal{F}^c \\ 0, & \text{otherwise} \end{cases} \quad \forall s \in S \tag{10}$$

$$x_s, \alpha^n_{f^s}, \beta^n_{f^s}, \delta^n_a \in \{0, 1\} \quad \forall n \in N, \forall s \in S, \forall f^s \in F^s \tag{11}$$

IV. Proposed Algorithm

The algorithm we propose to solve prioritized VNF-chain recovery problem is based on the greedy heuristic. The pseudo-code for the algorithm is presented in Alg. 1. The algorithm takes as an input the following: a list of all VNF-chains S, a list of non-broken chains $S' = \{s \in S : x_s = 1\}$), VNF assignments ($\alpha$) before failure, accelerator allocations (β) before failure and a set of all non-failed server-nodes (N) with their respective resource usages.

A sorted list S_b of all broken-chains in the order of their decreasing priorities, which is based on their μ_s values is created (1). Also, a list of all chains S_{rev} in increasing priority is created (2).

For accelerator-agnostic VNF-chain recovery, placement decisions are agnostic to the availability of accelerators. As a result, placement decisions are based solely on CPU resources and accelerator-allocation is done only if hardware-accelerator resources are available on that node. In other words, no explicit effort is made by the heuristic to allocate an accelerator to a VNF resulting in an inefficient resource-utilization. The algorithm for accelerator-agnostic VNF-chain recovery does not contain lines (10-16) of Alg. 1.

With accelerator-aware allocation, the decision for accelerator allocation is decoupled from other placement decisions. As opposed to the accelerator-agnostic placement, accelerator-aware placement makes use of `accelVNF` procedure for the explicit allocation of an accelerator to a VNF. The pseudo-code for `accelVNF` procedure is shown in Alg. 2. First, it is checked if enough resources are available on any server-node with an attached hardware-accelerator by using `AccAlloc` procedure. If not, a node with a hardware-accelerator is randomly selected (n_a) from a list (N_a) of server-nodes with hardware-accelerator sorted in the order of increasing available CPU. To accommodate f^s on n_a, CPU, bus or accelerator resources are made available by removing VNFs of the lowest priority VNF-chain if the priority of VNF-chain s is more than s_b (9-20). A VNF-chain is selected in-order from S_b for its restoration and the set of non-placed VNFs is assigned to F^b. For every VNF f^b which has an accelerator implementation available in A, its placement is first tried on a server-node with hardware-accelerator available using the procedure `accelVNF` (Alg. 2). If procedure `accelVNF` returns $None$, its placement is continued as usual.

If not enough CPU resources are available on any server-node, VNF-chains are removed sequentially in the greedy manner i.e. lower-priority chain first (19-26) from S_{rev}. The removed VNF-chain s_r is added to the sorted-list of the broken chain S_b and resources are updated using the `RemoveVNFs` procedure.

In the case when all VNFs of s_b are placed successfully, S' is updated and the next VNF-chain with the lower priority

Algorithm 1: VNF-chain recovery algorithm

```
Input  : $S$, $S'$, $\alpha$, $\beta$, $N$
Output: $S'$
$S_b \leftarrow$ sorted list of broken chains in decreasing priorities;
$S_{rev} \leftarrow$ sorted list of chains in increasing priorities;
$i_b, i_r \leftarrow 0$;
$l \leftarrow 0$;
while $i_b < |S_b|$ do
    $s_b \leftarrow S_b[i_b]$;
    $F_b \leftarrow \{f^s \in F^{s_b}$ : VNFs in $F^{s_b}$ which are not placed$\}$;
    $fail \leftarrow$ False;
    for $f^b$ in $F_b$ do
        if atype($f^b$) $\in A$ then
            $n_a \leftarrow$ accelVNF($f^b$, $N$);
            if $n_a \neq$ None then
                $\alpha[f^b], \beta[f^b] \leftarrow n_a, n_a$;
                break;
            end
        end
        $n_p \leftarrow \arg\max_{n \in N} \mathcal{R}_{cpu}(n)$ ;
        while $cpu_0(f^b) > \mathcal{R}_{cpu}(n)$ and $i_r \leq |S_{rev}|$ do
            $s_r \leftarrow S_{rev}[i_r]$;
            if ChainPrior($s_b$)>ChainPrior($s_r$) then
                RemoveVNFs($F^b$);
                $n_p \leftarrow \arg\max_{n \in N} \mathcal{R}_{cpu}(n)$ ;
                $i_r \leftarrow i_r + 1$;
            else
                break;
            end
        end
        if $cpu_0(f^b) > \mathcal{R}_{cpu}(n_p)$ then
            $\alpha[f^b] \leftarrow n_p$;
        else
            $fail \leftarrow$ True;
            break
        end
    end
    if $fail ==$ True then
        RemoveVNFs($F^b$);
    else
        $S' \leftarrow S' \cup s_b$;
    end
    $i_b \leftarrow i_b + 1$;
end
end
```

is considered for recovery. At the end of the algorithm S' contains all the VNF-chains which have all its VNFs restored.

V. Evaluation

The ILP model for accelerator-aware VNF-chain recovery problem is implemented in CPLEX (v12.9) framework using doCPLEX Python API [11] and the heuristic algorithm proposed in Alg. 1 is implemented using Python language. In this section, we describe the evaluations carried out in order to assess the efficiency of the ILP and the heuristic with regards to the VNF-chain recovery problem. First, the

Algorithm 2: VNF accelerator allocation procedure

```
Procedure accelVNF($f^s$, $N$):
    $N_a \leftarrow$ sorted list of nodes $N$ with accelerator in
      increasing CPU;
    for $n_a$ in $N_a$ do
        if AccAlloc($f^s$, $n_a$)$==$ True then
            return $n_a$;
        end
    end
    $n_a \leftarrow$ RandomChoice($N_a$);
    if $cpu_0(f^s) - cpu_r(f^s) > \mathcal{R}_{cpu}(n_a)$ then
        $s_b \leftarrow$ lowest priority chain with atleast one VNF
          placed on $n_a$.;
        if ChainPrior($s$)>ChainPrior($s_b$) then
            RemoveVNFs($\{f^{s_b} \in \mathcal{F}^{s_b} : \alpha[f^{s_b}] = n_a\}$);
        end
    end
    if $2t_s > \mathcal{R}_{cpu}(n_a)$ or r(atype($f^s$)) $> \mathcal{R}_{acc}(n_a)$
      then
        $s_b \leftarrow$ lowest priority chain with atleast one VNF
          allocated accelerator on $n_a$.;
        if ChainPrior($s$)>ChainPrior($s_b$) then
            RemoveVNFs($\{f^{s_b} \in \mathcal{F}^{s_b} : \alpha[f^{s_b}] = n_a\}$);
        end
    end
    if AccAlloc($f^s$, $n_a$)$==$ True then
        return $n_a$;
    else
        return None;
    end
end
```

placement and accelerator allocation of VNFs is performed by using the heuristic which we proposed in [10]. The result of this heuristic is used as an input allocation for both the ILP and heuristic. A fixed number of server-nodes are chosen at random from all the server-nodes to cause the failure. In order to highlight the impact of accelerator-allocation criterion on the traffic-restoration, we compared the performance of accelerator-agnostic and accelerator-aware heuristics.

These evaluations were carried on a machine with Intel Xeon CPU and 16GB of memory running Ubuntu 16.04. Table II describes the value (or range) of various parameters involved in evaluations of ILP and heuristic.

TABLE II
Description of parameters and decision variables

Parameter	Value or range	Parameter	Value or range		
$\mid S \mid$	15, 150	$c_o(f^c)$	3-5		
$R_{cpu}(n)$	20-28	$c_i(f^c)$	(0.40 - 0.60)$c_o(f^c)$		
$R_{acc}(n)$	0,1	f_{acc}^{vnf} , f_{acc}^{fn}	0.20		
$R_{bus}(n)$	80-120 (Gbps)	accel. type	a_1	a_2	a_3
Chain length	4	accel. size	0.40	0.28	0.30
t_s	1.0-5.0 (Gbps)	Traffic-class	I	II	III
		Prob.	0.10	0.20	0.70

A. Execution time

The comparison of maximum execution times for the ILP model and heuristic method with a single-node failure is shown in Table III. As expected, the computational time for ILP model increases rapidly with the number of VNF-chains. This becomes an issue, especially for the case when the total number of VNF-chains is greater than 15 and execution time for ILP becomes orders of magnitude higher than that of heuristic. Therefore, solving VNF-chain recovery for problem instances of large-sizes becomes infeasible using the ILP model. However, the heuristic method can be used for larger problems as discussed in the next section.

TABLE III
COMPARISON OF MAXIMUM EXECUTION TIME FOR ILP MODEL AND HEURISTIC ALGORITHM

Total chains ($\mid S \mid$)	ILP time	Heuristic time
10	230 ms	1.3 ms
15	350 ms	1.8 ms
20	$>$ 100 s	3 ms

B. ILP and Heuristic comparison

Here, we compare the average amount of traffic lost following node failures for each priority-class without any traffic-recovery scheme with the amount of traffic lost after running the ILP-based traffic recovery model and heuristic approach. The comparison of traffic-lost for the single-node and multi-node failure is depicted in Fig. 1 and 2 with an input of 15 chains.
As expected an exact approach like ILP always has the minimum amount of lost-traffic. However, it can be observed that the performance of the heuristic is close to the ILP method. For VNF-chains of high and medium priorities amount of traffic lost is equal with ILP and heuristic approach. The amount of traffic lost for the lowest-priority with the heuristic is not more than 15% as that of ILP in all cases. Moreover, the proposed heuristic can be be used to solve the problem instances of large-sizes.

C. Impact of accelerator allocation criterion

In order to highlight the impact of accelerator-allocation on the amount of lost-traffic, we compare the performance of accelerator-agnostic heuristic with our heuristic as depicted in Fig. 3 with an input consisting of 150 chains.
As the higher-priority VNF-chains are restored equally by both ILP and heuristic, we focused only on the lowest-priority traffic. It can be observed that with the accelerator-aware heuristic about 30% of more traffic can be restored as compared to the accelerator-agnostic heuristic. This can be explained by the fact that a better VNF consolidation is achieved with the accelerator-aware heuristic in contrast with the accelerator-agnostic heuristic.

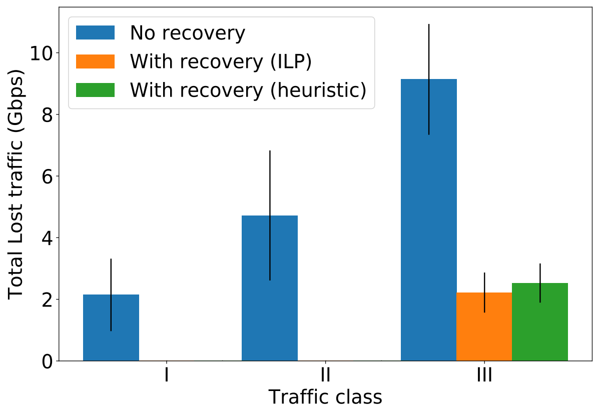

Fig. 1. Performance comparison of ILP and heuristic for a single node failure in terms of lost traffic.

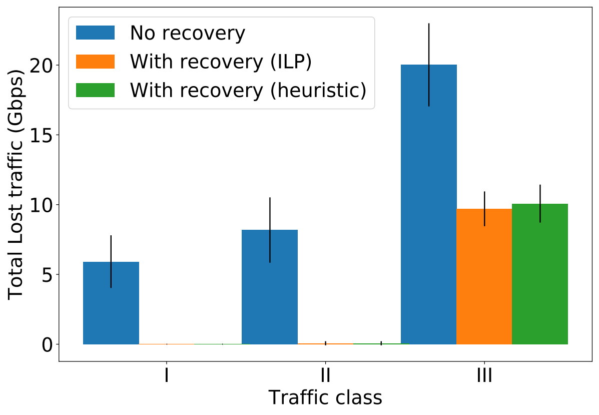

Fig. 2. Performance comparison of ILP and heuristic for multi-node (three) failure in terms of lost traffic.

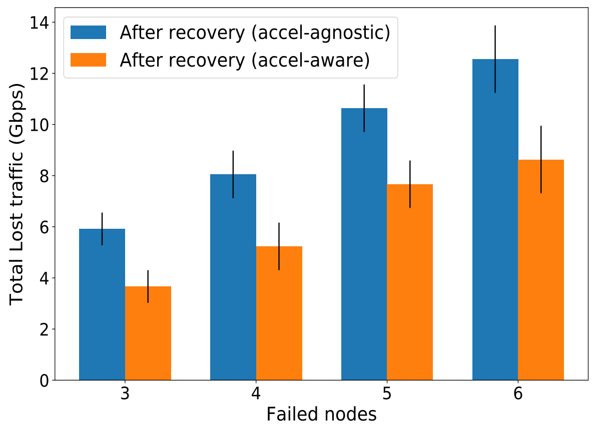

Fig. 3. Impact of accelerator allocation on the heuristic performance in terms of lost traffic.

VI. Conclusion

Advantages of integrating hardware-accelerators in NFVi includes performance speed-ups and cost savings due to the reduction in VNF CPU usage. In order to minimize losses incurred to telcos due to failures in heterogeneous NFV environments, VNF-chain recovery models need to be modified. This paper presented an ILP formulation to model accelerator-aware, prioritized VNF-chain recovery problem.
As the execution time for exact-methods like ILP is high, a greedy-based heuristic is introduced to solve larger instances of this problem. The performance of the heuristic is on a par with ILP as far as the restoration of high and medium priority VNF-chains is concerned, with a slight penalty only for low-priority VNF-chains. The evaluation also shows that the performance of accelerator-aware heuristic is 30% better than the accelerator-agnostic heuristic in terms of the total amount of traffic lost.
For future work, we would investigate the availability of VNF-chains hardware-accelerators along-with usual reliability parameters like VNF and server-node reliabilities.

VII. Acknowledgment

This work was funded through NGPaaS, under the grant number 761557, in the scope of the European Commission Horizon 2020 and 5G-PPP programs.

References

[1] L. Linguaglossa, S. Lange, S. Pontarelli, G. Rétvári, D. Rossi, T. Zinner, R. Bifulco, M. Jarschel, and G. Bianchi, "Survey of performance acceleration techniques for network function virtualization," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 746–764, 2019.

[2] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. IEEE, 2015, pp. 171–177.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 50–56.

[4] H. Fan, Y. Hu, S. Zhang, and Q. Ren, "Hardware acceleration resource allocation mechanism for vnf," *Procedia computer science*, vol. 131, pp. 746–755, 2018.

[5] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 128–133.

[6] A. Tomassilli, N. Huin, F. Giroire, and B. Jaumard, "Resource requirements for reliable service function chaining," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[7] F. Carpio and A. Jukan, "Improving reliability of service function chains with combined vnf migrations and replications," *arXiv preprint arXiv:1711.08965*, 2017.

[8] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, "Dynamic hardware-acceleration of vnfs in nfv environments," in *2019 Sixth International Conference on Software Defined Systems (SDS)*. IEEE, 2019, pp. 254–259.

[9] S. Dräxler and H. Karl, "Spring: Scaling, placement, and routing of heterogeneous services with flexible structures," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 115–123.

[10] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, "VNF-AAP: Accelerator-aware Virtual Network Function Placement," Sep. 2019, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02292930

[11] IBM, "IBM ILOG CPLEX optimization studio." [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer