On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge

Amir Mohamad
School of Computing
Queen's University
Kingston, Ontario, Canada
Email: {a.mohamad}@cs.queensu.ca

Hossam S. Hassanein
Schoole of Computing
Queen's University
Kingston, Ontario, Canada
Email: {hossam}@cs.queensu.ca

***Abstract*—The demand for edge resources is increasing and will continue to rise especially because of delay-sensitive applications. Because of the limited resources at the network edge, efficient resource utilization will play a crucial role. In this paper, we demonstrate the gain of VNFs sharing-based service function chaining (SFC) requests placement, as a way of satisfying more requests with average less resources per request. We formulated the sharing-based SFC placement as an integer linear program (ILP) to minimize the overall deployment cost, hence optimize resource utilization and yet satisfy the QoS requirements. Our experiments show that sharing deployed underutilized VNFs will help satisfy 9-47% more SFC requests with on average 14-46% less resources per request.**

***Index Terms*—NFV, VNF placement, Service Function Chaining, edge computing, MEC**

I. Introduction

The next generation of mobile networks (5G) is expected to address performance requirements of divers use cases in different vertical industries. Massive machine type communications (mMTC), enhanced mobile broadband (eMBB), and ultra-reliable low-latency communications (URLLC), are the categories of addressed use cases. To fulfill the requirements of these diverse use cases, innovations and enhancements are needed on the radio side and the networking (core) side. On both sides, network function virsualization (NFV) [1], software-defined networking (SDN), and edge computing are to play a principal role [2].

All services, including network services, consist of component functions that are stitched together in a specific order to form service function chains (SFCs). NFV brings elasticity to the creation and deployment of services by virtualizing their functional components and decoupling them from their specially-built hardware [3].

A virtualized infrastructure of telco clouds extending from the network core to the perimeter of the access network, forms a layer that can be used to host delay-sensitive network services and applications of service providers. The layer could even be extended to the cell-sites in the radio access network (RAN) to host virtualized base-band units (BBUs) as well as other third-party workloads. The part of telco cloud that extends from access central offices (COs) to the RAN is called the edge. With the growing interest in delay-sensitive applications, such as AR/VR, telehealth, online gaming, autonomous vehicles, and content delivery services, the demands put on the edge part of telco clouds will be high and no doubt will grow significantly after the field deployment of 5G networks.

In mobile/multi-access edge computing (MEC) as the mobile networks version of edge computing, hosted services will have access to user mobility data as well as wireless channel related measurements. With such visibility, service providers will be able to take actions to enhance both the quality of service (QoS) and users' perceived quality of experience (QoE) at a millisecond scale. Having access to such information would not be viable anywhere else, that is why network edge especially MEC is regarded as the monetization arm that telecommunications service providers (TSPs) will utilize for profit and to cover the cost of upgrading.

Network edge has limited resources compared to the core cloud. Considering the anticipated high demand for edge resources and the importance of the edge being a precious asset for TSPs, the efficient utilization of edge resources will play a significant role in the fulfillment of delay-sensitive services and applications requirements.

In TSP telco clouds, virtual network functions (VNFs) that are assigned all required compute and bandwidth resources, are expected to operate at its full capacity. However, due to changing operation conditions, some VNFs might be underutilized, that is to receive and process traffic less than its full capacity. With SFC requests continuously arriving, it would be more resource-efficient to utilize deployed underutilized VNFs first, and only deploy a new VNF instance if no deployed underutilized VNF of the same type already exists. All this should be done with the SFC performance requirements, end-to-end latency in our case, in mind.

To this end, in this paper, we demonstrate the gain in efficient resource utilization and number of satisfied SFC requests by sharing VNFs across different SFCs. The main contribution of this paper is the introduction of VNF sharing-based SFC placement. We formulated the SFC placement and sharing as an integer linear program (ILP) model to demonstrate the VNF sharing gain. The objective is to minimize the

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: RGPIN-2019-05667.

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

total deployment cost, hence optimize resource utilization and still satisfy QoS requirements/constraints. Also, we compared number of satisfied requests and average required resources per SFC request of our sharing-based placement against no-sharing placement.

The remainder of this paper is structured as follows. In section II, we present related work. Proposed sharing-based SFC placement, network model, VNFs, SFCs and problem formulation are detailed in section III. Performance evaluation, simulation settings, experiments and results will be covered in section IV, section V concludes the paper.

II. Related Work

With the introduction of NFV in 2012 [1], service provisioning became more agile and placement of service functions/VNFs as the building block of SFCs started to gain traction. The wheel was not reinvented, researchers started by building on already existing body of research on cloud computing, virtual machine (VM) placement (VMP) and virtual network embedding (VNE). Due to the differences between VNFs placement and VMP [4], more research into the area was still needed. SFC placement is a two-step process: first, resource allocation of VNFs; second, the traffic steering/path selection. Since 2012, considerable research work was conducted, some are generic VNF/SFC placement [5]–[14], while others are more into specific settings and use cases, like VNF placement at the edge-central cloud and service placement and replication in 5G edge [15]–[19]. The main goal in [15] is to minimize both end-to-end delay and deployment cost of mission critical delay-sensitive service chains, and the SFC placement is formulated as a MIP and further approximated using tabu search algorithm. The work in [16], proposes VNF placement on edge-central cloud in away that optimizes resource utilization and satisfies QoS requirements using analytic queuing and MILP models. In [19], authors propose optimizing the QoS of cloud RAN (C-RAN) by dynamically configuring remote radio heads (RRHs) to proper BBU sectors according to the varying traffic conditions. For further details on VNF placement, [20] is a recent survey on VMP and VNF placement.

In the majority of papers surveyed, the VNF/SFC placement is formulated as an integer programming (ILP, MIP or MILP) model. After demonstrating the bottom line performance, a more practical heuristic placement algorithm is then presented. Sharing of VNF by more than one SFC flow is triggered by the fact that some VNFs could be shared by SFC flows, such as anti-virus. With the exception of [5], none of the surveyed papers considered sharing deployed underutilized VNFs while deploying new SFC requests. The work in [5], proposes sharing VNF among SFC flows based on a predefined number of flows that a VNF can handle. The fixed number of flows a VNF can handle doesn't reflect the changing operation conditions and will still leave some VNFs underutilized. To the best of our knowledge, the sharing/utilization of already deployed underutilized VNFs based on the currently unused capacity, was never proposed as a way of satisfying more SFC requests with less resources and yet satisfy the performance/QoS requirements.

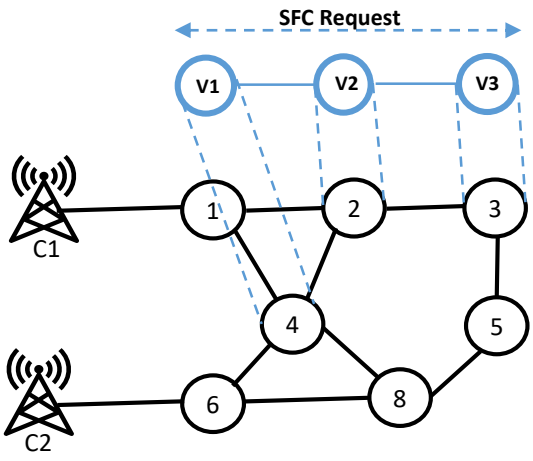

Fig. 1: Substrate network and SFC request placement.

III. Proposed Placement Technique

In our sharing-based placement and upon the arrival of SFC request, the priority is to use already deployed underutilized VNFs of the same type as those in current SFC request. At some point in the future, currently underutilized VNFs will get fully utilized even without hosting guest SFCs; hence, any sharing mechanism should take this into consideration. However, for demonstrating the benefits of sharing-based placement, we assume that currently underutilized host VNFs will remain so until the guest SFCs are concluded. We do not specify explicit start and termination points for SFC requests, rather, first and last VNFs of each SFC are regarded as the source and destination, respectively. Moreover, we assume the existence of an SDN controller (SDN-C) which will take care of configuring forwarding plane switches to forward traffic according to the SFC selected path. An SFC mechanism is assumed to be used in the TSP service domain, either using network service header (NSH) or SDN-C bump-in-the-wire technique that uses port-pairs, port-pair groups, and port-chains to configure SFCs. The latter is used in OpenStack Tacker and Open Virtual Networking (OVN).

In the rest of this section we will explain substrate network model and problem formulation.

A. Substrate Network Model

The substrate network is modeled as a graph $G(N, E)$, where N is the set of nodes and E is the set of links. To be more generic, we modeled all our network nodes to be capable of hosting VNFs. The links between nodes are directional, as shown in Fig. 1, a link between nodes 1 and 2 means there are two links, one link from node 1 to node 2 and another link from node 2 to node 1.

Each node has its own compute resources, CPU cores and RAM. Each link has its bandwidth capacity as well as its propagation delay, which is a function of its length. When created, resources amounts are assigned randomly to

TABLE I: Substrate network parameters

Parameter	Description
N	Set of substrate network nodes
E	Set of substrate network links
$cpu_c(n)$	CPU capacity in cores of node $n \in N$
$ram_c(n)$	RAM capacity in GBs of node $n \in N$
$cpu_{av}(n)$	Available CPU cores at node $n \in N$
$ram_{av}(n)$	Available RAM GBs at node $n \in N$
$L_{nn'}$	A link exists from node n to node $n^{'}$, $n, n^{'} \in N$
$bw_c(L_{nn'})$	BW capacity in Mbps of link $L_{nn'}$
$bw_{av}(L_{nn'})$	Available BW at link $L_{nn'}$
$Del(L_{nn'})$	Propagation delay of link $L_{nn'}$

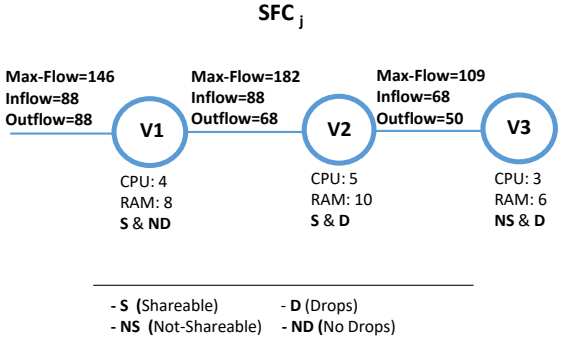

Fig. 2: VNFs of sfc_i and their required resources, max-flows, inflows and outflows

nodes and links. The substrate network topology is fixed and described by a connectivity matrix. The description of substrate network parameters is provided in Table I.

B. VNFs and SFC requests

An SFC request consists of an ordered list of VNFs. VNFs are selected from a list V of available already on-boarded VNFs. In the list of available VNFs, each VNF has a type, CPU and memory requirements, and the maximum traffic flow it can handle if assigned the resources required. Some VNFs, like firewalls, drop packets, in that case outflow should reflect such dropping. The outflow will be equal to inflow if a VNF does not drop or compress the inflow. As mentioned previously, some VNFs can be shared among SFC flows while others not. $S(v_i)$ is a flag to determine whether VNF v_i is shareable or not. Take for example request sfc_j shown in Fig. 2, it consists of 3 VNFs. The resources required as well as other parameters of each VNF are shown (real numbers from our simulation). As we can see the max-flow is proportional to the resources assigned to VNFs. For example v_2, v_1 and v_3 require 5, 4, and 3 CPU cores and can handle 182, 146 and 109 max-flows, respectively. Once selected in sfc_j, the inflow and outflow of VNFs should be determined. As shown in Fig. 2, the inflow of VNF v^j_{i+1}, $F_{in}(v^j_{i+1})$, is the out flow of preceding VNF v^j_i, $F_{out}(v^j_i)$, for all $v_i \in sfc_j$. Description of VNF and SFC request parameters is in Table II.

TABLE II: VNF and SFC request parameters

Parameter	Description
V	Set of available/on-boarded VNFs
v_i	Is a VNF, where $v_i \in V$
$cpu(v_i)$	CPU cores required for VNF $v_i \in V$
$ram(v_i)$	RAM GBs required for VNF $v_i \in V$
$F_{max}(v_i)$	Maximum inflow VNF v_i can handle
$S(v_i)$	Flag to indicate VNF v_i is shareable
$Drop(v_i)$	Flag to indicate that VNF v_i drops/compresses inflow
sfc_j	SFC request j
$\|sfc_j\|$	Number of VNFs in sfc_j
v^j_i	The i^{th} VNF of sfc_j
$F_{in}(v^j_i)$	Actual inflow that VNF v_i will be serving
$F_{out}(v^j_i)$	Outflow VNF v will produce
$Del(sfc_j)$	Maximum end-to-end delay of sfc_j

TABLE III: Decision variables and Constants

Variable	Description
X^j_{in}	Binary decision for placing VNF v_i of sfc_j at node n
R^j_{in}	Binary decision for sharing the flow of VNF v_i of sfc_j with already deployed VNF of same type at node n
D^i_n	VNF of same type as v_i already deployed at node n
$F_{av}(D^i_n)$	Available unused flow of v_i at node n
$U_c(cpu)$	Unit cost of cpu at all nodes
$U_c(ram)$	Unit cost of ram at all nodes
$U_c(bw)$	Unit cost of bw at all links

C. Problem Formulation

We formulated our problem as an ILP model, where all decision variables are binary and some constraints are quadratic. With SFC request sfc_j consisting of VNFs $v_i, i \in [1-|sfc_j|]$, our decision variables are; X^j_{in} if equals to one means a new instance of VNF v_i of sfc_j is to be placed at substrate node n and R^j_{in} means that VNF v_i of sfc_j is to share its traffic flow with already deployed underutilized VNF of the same type at substrate node n. Decision variables and other parameters descriptions are in Table III.

1) Objective Function: The objective is to select the placement that minimizes the overall cost, hence optimize resource utilization. The cost of instantiating a new instance of VNF v^j_i includes total required cpu, ram and bw costs. But when an already deployed VNF of the same type as v^j_i exists the cost only includes total required bw cost. The objective function in equation (1) is formulated in a way to favor sharing over instantiating and placing new VNF instances. The first term represents the cost of compute resources in case of deploying a new instance of VNFs. The second term is for the cost of bandwidth either in the case of sharing or the case of deploying a new VNF instance.

$$min \sum_{i=1}^{|sfc_j|} \sum_{n \in N} [cpu(v_i^j)U_c(cpu) + ram(v_i^j)U_c(ram)]X_{in}^j + F_{out}(v_i^j)U_c bw[X_{in}^j + R_{in}^j] \quad (1)$$

2) Constraints: Constraints are needed to assure that our model will converge to a feasible solutions. A feasible solution has to have each VNF of an SFC request mapped only once to a physical node. Moreover, the mapping should be either to place a new instance or to share already deployed instance, see (2).

$$\sum_{n \in N} X_{in}^j + R_{in}^j = 1 \quad , \quad \forall i \in [1 - |sfc_j|] \quad (2)$$

Constraints (3) and (4) ensure that, when a sharing decision is to be taken, there has to be an already deployed shareable VNF of the same type as the one in hand and there is enough available/unused flow that is enough for current VNF inflow.

$$\sum_{n \in N} X_{in}^j + D_i^j \, S(v_i) \, R_{in}^j = 1, \quad \forall i \in [1 - |sfc_j|] \quad (3)$$

$$F_{in}(v_i^j) \, R_{in}^j \leq F_{av}(D_n^i), \quad \forall i \in [1 - |sfc_j|] \quad (4)$$

For the placement decision X_{in}^j to be valid, there has to be enough cpu and ram resources at node n, constraints (5) and (6) ensure the availability of such compute resources.

$$\sum_{i=1}^{|sfc_j|} cpu(v_i^j) \, X_{in}^j \leq cpu_{av}(n), \quad \forall n \in N \quad (5)$$

$$\sum_{i=1}^{|sfc_j|} ram(v_i^j) \, X_{in}^j \leq ram_{av}(n), \quad \forall n \in N \quad (6)$$

For any two consecutive VNFs v_i^j and v_{i+1}^j of sfc_j to be placed on two nodes n and $n^{'}$: first, there has to be a link $L_{nn'}$ connecting the two nodes, constraint (7); second, the outflow of first VNF $F_{out}(v_i^j)$ should not exceed available bandwidth at that link $bw_{av}(L_{nn'})$, constraint (8). In both constraints, the two terms $(X_{in}^j + R_{in}^j)$ and $(X_{(i+1)n'}^j + R_{(i+1)n'}^j)$ represent the four possible placement decisions our model might take. First decision, to deploy new instances of v_i^j and v_{i+1}^j at nodes n and $n^{'}$, respectively. Second, to share already deployed instances of v_i^j and v_{i+1}^j at nodes n and $n^{'}$. Third, deploy a new instance of v_i^j at node n and share a deployed instance of v_{i+1}^j at $n^{'}$. Fourth, share a deployed instance of v_i^j at node n and deploy a new instance of v_{i+1}^j at node $n^{'}$.

$$L_{nn'} \, (X_{in}^j + R_{in}^j)(X_{(i+1)n'}^j + R_{(i+1)n'}^j) = 1 \quad \forall n, n^{'} \in N \; \& \; \forall i \in [1 - (|sfc_j| - 1)] \quad (7)$$

$$\sum_{n \in N} \sum_{n' \in N} F_{out}(v_i^j)(X_{in}^j + R_{in}^j)(X_{(i+1)n'}^j + R_{(i+1)n'}^j) \leq bw_{av}(L_{nn'}), \quad \forall i \in [1 - (|sfc_j| - 1)] \quad (8)$$

Finally, the performance requirements (end-to-end latency) of sfc_j must be satisfied. Even though end-to-end latency has many components such as processing delay, queuing delay, propagation delay and virtualization delay [11], for simplicity, the only component we opted for is the fixed propagation delay, see constraint (9).

$$\sum_{i=1}^{|sfc_j|-1} \sum_{n \in N} \sum_{n' \in N} Del(L_{nn'}) \, (X_{in}^j + R_{in}^j) \, (X_{(i+1)n'}^j + R_{(i+1)n'}^j) \leq Del(sfc_j) \quad (9)$$

IV. Performance Evaluation

To evaluate and demonstrate the performance gain of sharing-based SFC placement technique, we developed a Java-based simulation environment. The simulation environment generates substrate network model, creates SFC requests, executes the placement decisions, and tracks the network model total utilization as well as other measurments. The ILP model is solved using the Gurobi solver [21]. All simulation experiments executed on Dell OPTIPLEX 9020 machine of Intel core i7@3.6 GHz, 16 GB RAM with Windows 10 Enterprise. We used the NSFNET network topology with 13 nodes and 32 directional links.

Each time a substrate network model is created, the topology is fixed, but the resources cpu, ram and bw are drawn randomly from the ranges $[8 - 64]\, cores$, $[16 - 128]\, GB$ and $[100 - 1000]\, Mbps$, respectively. Moreover, the link length, that determines propagation delay, is also random and drawn from the range $[50 - 1000]\, m$.

SFC requests are also generated randomly. The SFC request length, resource requirements of each VNF, each VNF shareability and if it drops/compresses inflow, and VNFs inflow and outflow, all these parameters are random and drawn from predetermined ranges, as follows:

- SFC length range $[2 - 10]\, vnfs$
- SFC end-to-end latency = $(|sfc_j| - 0.5) *$ average-link-delay
- VNF cpu range $[2 - 8]\, cores$
- VNF ram range $[4 - 16]\, GB$
- VNF $maxflow$ is a function of required/assigned cpu and ram
- VNF $inflow$ range $[0.15 * maxflow - maxflow]\, Mbps$
- If VNF drops/compresses, $outflow$ range $[0.4 * inflow - inflow]$. If not, $outflow$ = $inflow$

We chose to set $U_c(cpu) = 2.5$, $U_c(ram) = 1.7$, $U_c(bw) = 2$, these arbitrary values have no effect on the results as they are the same on both sides of any comparison.

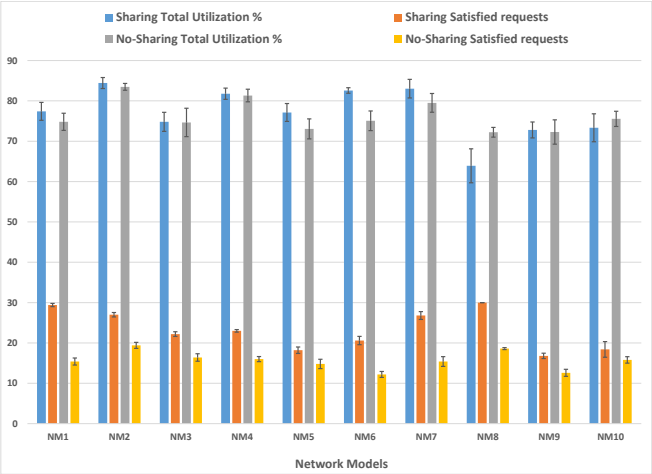

Fig. 3: Average total utilization and Average satisfied SFC requests (out of 30 requests) per network model. Experiment for each network model is repeated 5 times and the averages are presented

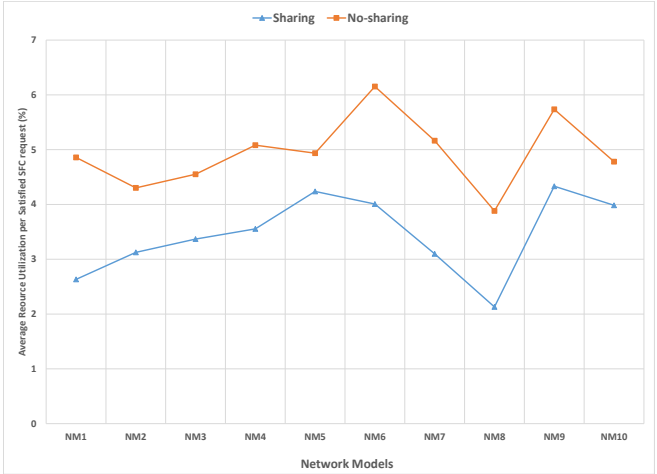

Fig. 4: Average percentage of required resources per SFC request

A. Experiments

For the purpose of demonstrating the gain, more requests satisfied with less average required resources per request, we designed two experiments. In the first experiment, we compared the proposed VNF sharing-based placement versus VNF no-sharing-based placement. In the second experiment, we studied the impact of increasing number of shareable VNFs in the list of available VNFs, hence number of deployed shareable VNFs.

1) Sharing vs No-Sharing: In this first experiment, 10 random network models are generated. For each network model an identical copy is cloned, one is used to satisfy SFC requests with shareable VNFs and the other without shareable VNFs. For each identical pair of network models, 30 SFC requests are generated of the same VNF list. Each SFC request is cloned and VNFs of the clone is set to non-shareable, one used for sharing-based placement and the clone for no-sharing-based placement, respectively. Each network model pair trial is repeated 5 times and the averages of the number of satisfied SFC requests and closing cpu utilization are recorded. Since the amount of ram resources is double the cpu resources both in substrate nodes and VNFs, we will only report and compare the cpu utilization. The bw utilization is ignored as it will be the same for sharing-based and no-sharing-based placement. As the results reveal, in Fig. 3, sharing the allocated capacity of already deployed VNFs, resulted in significant increase in number of satisfied SFC requests ranging from 9% to 47%, yet, as expected, using less compute resources. That is, accepting the same number of requests, the required resource are lower for sharing-based placement compared to no-sharing-based placement.

To show the average percentage of required/utilized resources per satisfied SFC request, in each network model trial we divided the closing utilization by number of satisfied SFC requests. As shown in Fig. 4, on average, the required/utilized resources per request for sharing-based placement is 14% to 46% less compared to no-sharing-based placement.

2) Number of Shareable VNFs (70%-based vs 30%-based): In this second experiment we study the impact of the number of shareable VNFs in the list of available VNFs. To do so, we created one typical pair of network models and we created a typical pair of available VNFs list. The first list is with 70% of its VNFs shareable and the other list is with 30% shareable VNFs. We used the two VNF lists to generate two sets of SFC requests, each with 30 requests. Peer SFC requests in the two sets are of the same length. For each network models pair, we repeated this experiment 10 times and reported the averages. The total number of accepted requests and the resources utilized are shown in Fig. 5. We observed that number of shareable VNFs impacts number of satisfied request and the required resources. Over the 10 experiments, number of accepted SFC requests is 13% to 26% higher for the 70%-based requests than for the 30%-based requests. On the other hand, the utilized resources of the 70%-based accepted requests range from 11% lower to only 5% higher than the 30%-based requests. When the same number of requests accepted, the amount of resources utilized is 11% less for the 70%-based requests than for the 30%-based requests.

V. Conclusion and Future Work

To take advantage of the changing operation conditions and the fluctuation in traffic flow over different periods, in this paper, we demonstrated the performance gain of sharing-based SFC requests placement. The demonstrated gain is in the form of increased number of satisfied SFC requests and a reduction of resources required to satisfy these requests. Indeed, the shared-based placement has to be done with current network

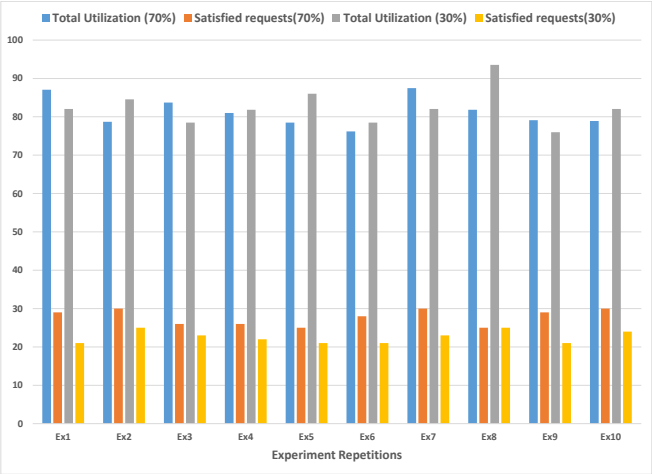

Fig. 5: Comparison of closing utilization and satisfied SFC requests (out of 30 requests) between 70%-based shareable VNFs and 30%-based shareable VNFs

load, and individual deployed VNFs' available flow in mind. Unlike previous related work, our sharing-based placement takes the decisions by considering current utilization status of deployed VNFs not based on a predetermined number of SFC flow a VNF can handle. The latter may still leave some VNFs underutilized. Our findings will help TSPs efficiently utilize their edge resources. This gain will translate into more earnings and better users satisfaction due to more satisfied requests/less blocked requests.

As a next step, we plane to consider different service categories. For example, a premium category service components shouldn't be shared even if underutilized. Also, we will incorporate more stochastic components such as, the expected time window a deployed VNF will remain underutilized, the expected available flow during such window, and the expected SFC requests arriving during the same window.

REFERENCES

[1] ETSI, "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," ETSI, Introductory White Paper, October 2012.

[2] F. van Lingen, M. Yannuzzi, A. Jain, R. Irons-Mclean, O. Lluch, D. Carrera, J. L. Perez, A. Gutierrez, D. Montero, J. Marti, R. Maso, and a. J. P. Rodriguez, "The unavoidable convergence of nfv, 5g, and fog: A model-driven approach to bridge cloud and edge," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 28–35, Aug 2017.

[3] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, "Service function chaining use cases in mobile networks," IETF, Internet Draft, January 2019.

[4] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "Vnf-fg design and vnf placement for 5g mobile networks," *Science China Information Sciences*, vol. 60, no. 4, p. 040302, Mar 2017. [Online]. Available: https://doi.org/10.1007/s11432-016-9031-x

[5] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*. IEEE, 2014, pp. 7–13.

[6] T. Kim, S. Kim, K. Lee, and S. Park, "A qos assured network service chaining algorithm in network function virtualization architecture," in *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, pp. 1221–1224.

[7] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

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

[9] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Network*, vol. 30, no. 3, pp. 81–87, 2016.

[10] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Computer Networks*, vol. 114, pp. 95 – 110, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617300087

[11] D. B. Oljira, K. Grinnemo, J. Taheri, and A. Brunstrom, "A model for qos-aware vnf placement and provisioning," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–7.

[12] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[13] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 4, pp. 1562–1576, 2018.

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

[15] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "Vnf placement optimization at the edge and cloud ," *Future Internet*, vol. 11, no. 3, 2019. [Online]. Available: http://www.mdpi.com/1999-5903/11/3/69

[16] F. Ben Jemaa, G. Pujolle, and M. Pariente, "Qos-aware vnf placement optimization in edge-central carrier cloud architecture," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.

[17] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[18] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5g edge network," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[19] M. Khan, R. S. Alhumaima, and H. S. Al-Raweshidy, "Qos-aware dynamic rrh allocation in a self-optimized cloud radio access network with rrh proximity constraint," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 730–744, 2017.

[20] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, 2018.

[21] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com