Virtual-Network-Function Placement For Dynamic Service Chaining In Metro-Area Networks

Leila Askari, Ali Hmaity, Francesco Musumeci, Massimo Tornatore
Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy
E-mail: firstname.lastname@polimi.it

***Abstract*—The advent of new services with stringent requirements on bandwidth and latency has led to a downward curve in per-user revenues of telecom operators. This has stimulated a significant shift in the way operators provision their services, moving from the utilization of dedicated and static hardware to support network functions (as NATs, firewalls, etc.), to the deployment of Virtual Network Functions (VNFs) in the form of dynamically-reconfigurable virtual machines on low-cost servers and switches. These VNFs must be chained together and should be placed optimally to meet the Quality of Service requirements of the supported services. This problem consists in placing the VNFs and routing traffic sequentially among them and is known as Service Chaining (SC). Solving this problem dynamically based on how traffic evolves allows to achieve great flexibility in resource assignment in the existing infrastructure and to save operational expenditures. An effective algorithm for dynamic SC must promote consolidation in VNF placement (it is desirable to consolidate VNFs in the fewer possible number of network nodes), while maintaining low blocking probability and guaranteeing latency targets to the supported services. In this paper we propose an algorithm which performs dynamic SC in a metro-area network, while minimizing average number of nodes required to host VNF instances as well as the blocking probability. This algorithm can help telecom operators reduce their operational expenditures up to 50% by activating less nodes to host VNFs in the network, while maintaining an acceptable level of blocking probability.**

I. Introduction

Cost-effectively provisioning new services (as Augmented Reality) that might require high bandwidth, or be latency sensitive and have higher reliability requirements is a complex challenge for operators. In response to this pressure, the concept of Network Function Virtualization (NFV) has attracted the attention of operators, as it enables to reduce Capital Expenditures (CapEx) and Operational Expenditures (OpEx) by virtualizing network functions (as NATs, firewalls, etc.) using virtual machines (VMs) running on top of standard servers and switches, by promoting resource sharing, and by decreasing energy consumption thanks to the consolidation of many network functions within few shared facilities. By chaining these Virtual Network Functions (VNFs) together (i.e, by placing VNFs and route traffic sequentially among them), the operator can provide a specific service (e.g., Cloud Gaming, VoIP, etc.) referred to as Service Chain (SC) [1] [2].

As NFV decouples network functions from hardware-based network appliances, network operators, based on the current situation of the network, in terms, e.g., of the amount of traffic and type of SC requested by users, can activate VNFs (by assigning resources to them) and deactivate them (by releasing resources used by them) in different network nodes equipped with processing units (identified to as "NFV-nodes"). In this context, to provision a SC it is important to decide in which network node to locate VNFs and how to route the traffic among them, as a proper placement of the VNFs can lead to efficient resource utilization and better Quality of Service (QoS). Hence, network operators should use efficient dynamic service chaining algorithms which, on one hand, help them reduce the expenses by activating less VNFs on less nodes and, on the other hand, minimize the blocking probability. Note that, by activating more VNF instances in the network, the network operator can serve more traffic, but (since activating an instance of a VNF imposes additional cost on network operators in terms of hardware resources, required licenses for softwares and power consumption among others) at the same time will face an increase in OpEx. Hence an appropriate trade-off must be investigated.

Considering specific features of a metro network in terms of latency, type of services requested by users, changing traffic load, number of users and services requested by users, network operators need efficient algorithms for VNF placement able to provision SCs dynamically based on current network condition. Most of existing studies on service chaining deal with static provisioning of SCs while the dynamic service chaining problem has received little attention so far [3] [4]. In this paper we provide an algorithm for dynamic VNF placement for SC provisioning in a metro network where at each time instant a number of users, request a specific SC and based on the current condition of the network VNFs are placed on NFV-nodes in a way that, with minimum possible number of provisioned VNFs, the blocking probability is minimized. The algorithm performs VNF placement in such a way that the bandwidth requirements of links, computational requirements of the NFV-nodes and latency requirements of requested SC are satisfied and wavelength continuity at each node is enforced. Moreover, by provisioning different SCs in the same wavelength, grooming is done to exploit maximum capacity of the network. Our algorithm is able to balance the trade-off between minimizing latency violation, decreasing blocking probability and reducing OpEx.

The remainder of this paper is organized as follows. Section II provides a brief overview of the related works. Section III, describes the metro network architecture and topology considered in this paper. The proposed heuristic algorithm is

978-3-903176-07-2

presented in Section IV. Numerical results are presented in Section V. Finally, conclusion is discussed in Section VI.

II. Related Work

The problem of VNF placement and traffic routing for SC provisioning has been subject of intense investigation in the last years, especially in static settings. For static SC provisioning, most studies propose an integer linear programming (ILP) model to obtain the optimal solution. For example authors in [5] dealt for the first time with a formal modeling of service chaining problem, and defined it as a Mixed Integer Quadratically Constrained Program (MIQCP) which finds the placement of VNFs and chains them together considering resource limitation of the network. Authors in [6] design a dynamic programming algorithm to jointly place VNFs and route traffic between them. They divide the problem in smaller subproblems and solve them sequentially [7]. Also some heuristic algorithms for VNF placement have been already proposed, as in [8]. However, in all the above-mentioned works, provisioning of SCs is performed under a static traffic assumption.

Dynamic placement of VNFs is addressed in a very limited set of works. In [1] the dynamicity is accounted by considering that type, number and location of VNFs traversed by a given user's data flow may change in time. So a traffic-conditioning function is proposed which, based on Service Level Agreement (SLA) of each user, decides the type of traffic conditioning function (shaping or priority scheduling) suitable for a given user's data flow. Authors of [9] provide an online algorithm for VNF scaling to dynamically provision network services in a datacenter network. In [10] a Mixed Integer Programming formulation and a heuristic algorithm are provided to dynamically provision SC, again in a datacenter network, where an appropriate resource management is done based on number of users requesting SCs. The authors of [3] consider dynamic SC provisioning for two types of users in the network, new users and existing users that can change location in the network and change their requested SC. So, they propose at first an ILP model for service provisioning with the objective to maximize the profit of the service provider; then, to reduce time complexity, they provide a more scalable model based on column generation [4]. However, no existing work provides an algorithm for dynamic SC provisioning in a metro network capable of addressing the trade-off between QoS requirements of services, blocking probability and CapEx and OpEx of telecom operators, as the one provided in this paper.

III. Metro network architecture and Background on service chain provisioning

The topology that we considered in this paper is a 4-level hierarchical metro aggregation network that connects cell sites (to which users are connected) to Central Offices (COs). Access COs provides the connectivity between cell sites connected to them and Core CO through Main COs. In this network optical transparent switches are used which impose wavelength continuity. As it is shown in Fig. 1 a SC can be considered as a chain of VNFs that are virtual nodes, chained together using virtual links (i.e. connections between nodes along the SC) forming SC's path [11].

To perform service chaining the VNFs need to be placed on NFV-nodes and virtual links need to be mapped to (a set of) physical links. Each SC is also characterized by source of the SC request, which in our topology is a cell site, and destination of SC request, which for the SC shown in Fig 1 is Core CO.

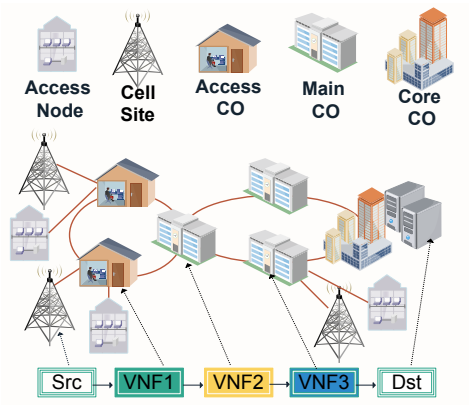


Fig. 1: Network topology.

To verify if latency requirements of SCs are satisfied, we model latency contributions as follows: i) context switching delay [12], that arises when a CPU is shared among multiple VMs and a delay is incurred for loading or saving the state of VMs. For each node n in NFV-nodes of SC ($n \in NFV_{SC}$), we consider a fixed value for context switching delay (DV_n) [13]; ii) propagation delay for each link l (DP_l) in SC's path ($Path_{SC}$); iii) Forward Error Correction (FEC) delay for (de)encoding optical signal in intermediate NFV-nodes of a SC; iv) optical-electrical-optical (OEO) conversion delay in intermediate NFV-nodes of a SC [14]. Overall, the end-to-end latency for a SC can be calculated as follows:

$$D_{SC} = \sum_{n \in NFV_{SC}} (DV_n + OEO_n + FEC_n) + \sum_{l \in Path_{SC}} DP_l$$

IV. Dynamic VNF placement Algorithm

In this section we describe the proposed dynamic algorithm, named Dynamic VNF Placement (DVNFP), that finds the placement for VNFs of a SC and route traffic between them at each time instant considering current state of the network.

A. Problem Statement

The problem can be stated, in a summarized form, as follows. We are given a hierarchical optical aggregation network which is composed of COs connected together using Wavelength Division Multiplexing (WDM) links. In this network, SC requests are dynamically generated by users (source of request) and they are generated in cell sites and terminated in a CO (destination of request) depending on the type of SC. Upon the arrival of a SC request, we need to decide the placement of its VNFs on NFV-nodes with the objective of

minimizing the number of activated VNF instances as well as blocking probability, constrained by maximum link capacity, maximum node computational capacity and maximum tolerable SC latency.

A SC request is characterized by a given holding time and number of users requesting that specific SC. In addition, each SC requires a specific amount of bandwidth and has a maximum tolerated latency. Furthermore, for any SC, the corresponding VNFs require a specific amount of computational capacity in terms of fraction of CPU core usage per user.

B. Algorithm

In order to provision a SC we need to place all of its VNFs on NFV-nodes and route traffic between them. DVNFP first builds an auxiliary graph of network with all the nodes and links with their wavelengths. It takes as input a SC request which is specified by these properties:

- *src*: source of the SC request
- *dst*: destination of the SC request
- N_{vnf}: number of VNFs composing the SC
- F_{sc}: type of VNFs composing the SC
- N_u: number of users requesting the SC
- L_{sc}: latency requirements of the SC
- H_{time}: holding time of the SC

The pseudocode related to the placement of VNFs is shown in *Algorithm*1. The main steps of DVNFP for VNF placement can be defined as follows:

- *Reusing active VNFs:* Since activating an instance of a VNF imposes additional cost on network operators, when a SC request arrives, DVNFP tries to reuse already activated VNF instances in the network as much as possible. Therefore, as it is shown in *Algorithm* 1 line 4, for each VNF, DVNFP first checks if there is an already activated VNF instance of the same type in the network or not.
- *Selecting among active VNFs:* As it is shown in *Algorithm* 1 lines 5-21, if more than one VNF instance with enough capacity is already activated in the network, DVNFP uses a metric called "locality-awareness". This metric is obtained by summing up the length of the shortest path between source of SC request and the selected NFV-node, and the shortest path between that node and destination of SC request. It is worth mentioning that we use an adaptive Dijkstra algorithm to calculate the shortest path, in which the congested links are not included in the graph. DVNFP chooses the NFV-nodes with locality-awareness metric lower than a predefined threshold δ whose value can be decided based on the topology of the network. Among these NFV-nodes, our algorithm based on the requested SC decides which node to choose. So, if a SC request requires large computational resources e.g., Cloud Gaming [15], (requirements of these services will be quantified in Section VI) then the NFV-nodes closer to the Core CO, which are more likely to have large computational capacity are chosen. However, if the SC has stringent latency requirements (e.g. as happens for Smart Factory), DVNFP tries to serve that SC locally, using as NFV-nodes access COs or at least CO in lower level of the metro hierarchy. When the best NFV-node is found the VNF is placed on that node by allocating the required computational capacity.
- *Activating new VNF instance:* If no VNF instances of a certain VNF are already activated in the network, DVNFP tries to instantiate a new one. As it is shown in *Algorithm* 1 lines 22-30 at first it calculates the shortest path between source and destination of the SC request. Then it tries to place the VNF on the closest NFV-node to the source along the shortest path with enough computational capacity. If the VNF cannot be placed on any of NFV-nodes along the shortest path, the algorithm checks the capacity of all other NFV-nodes on the network and tries to place the VNF on the node with better locality-awareness and higher betweenness centrality (defined as number of shortest path passing through this node).

Note that, at each step, source of SC request is replaced by the NFV-node chosen to host a VNF at previous step and the above-mentioned procedures are repeated until all the VNFs of a SC are placed.

The pseudocode of QoS improvement is shown in *Algorithm* 2. When all the VNFs are placed, as it is shown in *Algorithm* 2 line 2, algorithm checks if latency requirement of the SC is satisfied. If it is the case, the SC is provisioned and when its holding time expires the resources used by this SC (link capacity and computational capacity of NFV-nodes) are released. These steps are shown in *Algorithm* 2 lines 3-6. If the VNF placement does not allow to meet latency requirements, algorithm calculates the latency of all virtual links and finds the one with the highest value of latency. After that, the resources on the endpoints of this virtual link are released and their VNFs are placed on their adjacent virtual nodes (if they have enough computational capacity). Then the shortest path between these new endpoints is calculated and is replaced with that virtual link with the highest latency. This procedure is referred to as VNFs grouping and is shown in *Algorithm* 2 lines 8-17. In Fig. 2 we demonstrate how grouping of VNFs is done. In this example the virtual link between VNF2 and VNF3 is the one which has the highest value of latency. Therefore, we need to release computational resources allocated to VNF2 on node 3 and to VNF 3 on node 6 and place VNF2 and VNF3 on node 2 and node 7 respectively. When the VNFs are placed on the new NFV-nodes the required computational resources on those nodes are allocated to these VNFs and the shortest path between these two NFV-nodes is calculated. Then the calculated shortest path is calculated and considered as the new virtual link connecting VNF2 and VNF3 instances. DVNFP repeats the same procedure until either latency requirement of the SC is satisfied, or all the VNFs are consolidated.

Algorithm 1 Placement of Virtual Network Functions

Given: Service Chain request $Req(src, dst, N_{vnf}, N_u, F_{sc}, L_{sc}, H_{time})$, actual network state N_{state}
* *Phase I* *\
repeat
  if ∃ instance of VNF already placed **then**
    Select all the VNF instances.
    Sort NFV-nodes $f \in F_{li}$ where VNF instances are hosted by increasing value of loc_f and select only the VNF instances which satisfies: $loc_f - length(sp) < \delta$.
    if ∃ more than one such VNF instance **then**
      Choose the node with less activated VNF instances.
      if ∃ more than one such NFV-node **then**
        if SC requires computational capacity **then**
          Choose the NFV-node closer to Core CO.
        else
          Choose the NFV-node closer to src.
        end if
      end if
    end if
    Try to scale up the VNF instance in the selected NFV-nodes until success or all NFV-nodes have been tried.
    if success **then**
      update N_{state}
      continue
    end if
  else * *Find an NFV-nodes with enough capacity and place VNF* *\
    Select in order the NFV-nodes on the shortest path between src and dst of the SCs.
    Sort the NFV-nodes by increasing number of active VNFs on nodes.
    Try placing the VNF instance on an NFV-node until all the NFV-nodes on the shortest path have been tried.
    if failed **then**
      Select the NFV-node on the network with better loc_f and higher betweenness centrality.
      Try placing the VNF instance on an NFV-node until all the NFV-nodes have been tried.
      if failed **then**
        return *SC request blocked due to capacity*
      else
        Update N_{state}
      end if
    else
      Update N_{state}
    end if
  end if
until All the VNFs of the SC request are chained

Algorithm 2 QoS Improvement

* *Phase II* *\
Check end-to-end latency of the embedded SC against requirement.
if success **then**
  Provision SC request.
  Release the resources when H_{time} expires.
  return *SC request provisioned*
else
  repeat
    Select the virtual link with highest latency.
    Release the resources of the VNFs on its end-points.
    Find the two closest nodes to two end-points on SC virtual path with enough capacity.
    if Such node not found **then**
      return *blocked SC request due to latency.*
    else
      Enable those VNFs on these two nodes
      Add virtual link between those two nodes to SC virtual path.
    end if
    if End-to-end latency is satisfied **then**
      Provision SC request.
      Release the resources when H_{time} expires.
      return *SC request provisioned.*
    else if Consolidated all virtual links and latency not satisfied **then**
      return *blocked SC request due to latency.*
      Release all the resources provisioned earlier.
      Update N_{state}.
    end if
  until Latency satisfied or all VNFs have been consolidated
end if

C. Benchmark Algorithms

We considered two benchmark algorithms to evaluate the performance of DVNFP which are as follows:

- **Centralized service chaining:** In this approach we place all the VNFs in the node with highest computational capacity (Core CO in our topology) and we serve all the SCs using the VNF instances on that node.
- **Distributed service chaining:** In this approach we enable VNF instances on all the NFV-nodes whenever they are needed. In other words, even if there is already an activated instance of a VNF in the network, the algorithm enables a new instance on NFV-nodes along the shortest path between source and destination of the SC request. Algorithm repeats the same steps till all the VNFs are placed. If length of shortest path is less than number of VNFs that are needed to be placed to provision the SC, algorithm tries to put rest of VNFs on destination.

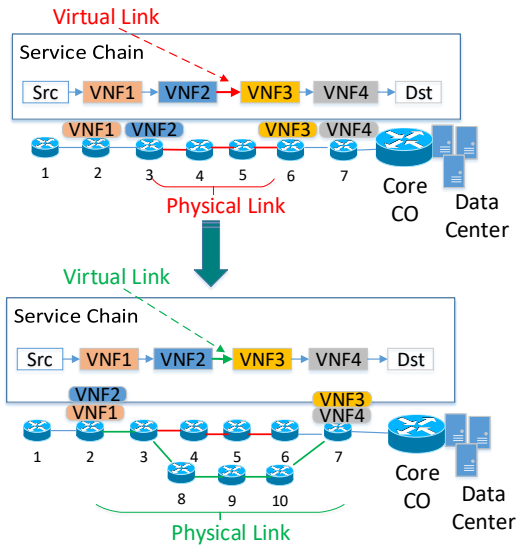


Fig. 2: VNF Grouping

V. Illustrative Numerical Results

In this section we compare the performance of DVNFP vs. the two benchmark cases, centralized service chaining and distributed service chaining. We use three performance metrics for comparison; blocking probability, which is calculated considering number of SC requests served out of total number of SC requests during the simulation, average number of active NFV-nodes, which is calculated considering number of NFV-nodes that have at least one running VNF instance at each time instant, and latency violation ratio which shows number of SC requests that violated latency requirements out of total SC requests.

Network modeling : We considered a topology similar to that shown in Fig.1 in which we have 80 nodes, 15 of which are NFV-nodes while the remaining nodes are forwarding nodes. The topology has 170 WDM links each supporting 16 wavelengths with 40 Gbit/s capacity. At each node wavelength continuity is enforced (we consider an optical network substrate), unless the node hosts a VNF, in which case, the intermediate virtual link is terminated and wavelength conversion is admissible. We assumed that for each SC request source is chosen randomly among cell sites while destination can be either Core CO or one of NFV-nodes based on the requested service type. Each NFV-node is equipped with 512 CPU cores, whereas the Core CO is assumed to have unlimited computational capacity.

Traffic/SC modeling: We conducted our simulative experiments using a C++ discrete-event driven simulator, that generates SC-requests as input traffic according to a Poisson-distribution of inter-arrival rates and negative-exponential distribution of the holding times (with average duration equal to one). All the plotted results have been obtained guaranteeing 95% statistical confidence and at most 5% confidence interval. We considered 6 different SC types as illustrated in Table I with different bandwidth and latency requirements [11] [15]–[18]. The VNFs are Network Address Translation (NAT), Firewall (FW), Traffic Monitor (TM), WAN Optimizer (WO), Video Optimization Controller (VOC), Intrusion Detection and Prevention System (IDPS). Each VNF requires specific amount of CPU resources per user. Table II illustrates the required amount of CPU (in terms of percentage of CPU per user) for each VNF [19]–[21].

TABLE I: Service Chains With Corresponding VNFs, Bandwidth and Latency Characteristics

Service Chain	Service Chain VNFs	Bandwidth	Latency
Cloud Gaming	NAT-FW-VOC-WO-IDPS	4 Mbps	80 ms
Augmented Reality	NAT-FW-TM-VOC-IDPS	100 Mbps	1 ms
VoIP	NAT-FW-TM-FW-NAT	64 Kbps	100 ms
Video Streaming	NAT-FW-TM-VOC-IDPS	4 Mbps	100 ms
MIoT	NAT-FW-IDPS	100 Mbps	5 ms
Smart Factory	NAT-FW	100 Mbps	1 ms

TABLE II: CPU Core Usage for VNFs

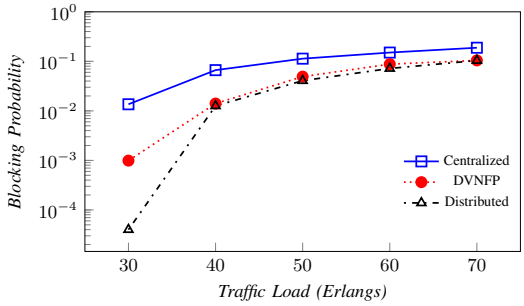
VNF Name	CPU Core Per User
NAT	0.00092
FW	0.0009
VOC	0.0054
TM	0.0133
WO	0.0054
IDPS	0.0107

A. Comparison Between Algorithms

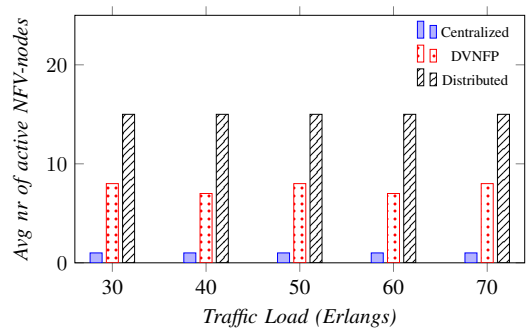
We compare the three algorithms for increasing traffic load values. Fig 3(a) shows the blocking probability increase for increasing load in the network. We notice how blocking probability for DVNFP always lies in between blocking probability of two benchmark algorithms i.e. centralized and distributed, returning, for most cases, and especially for higher loads, results very similar to the distributed case. This observation is very promising, as it confirms that our algorithm guarantees a blocking close to the blocking lower bound (i.e, the one returned by a completely distributed service chaining approach). Fig 3 (b) plots the average number of active NFV-nodes. In this case we can see that DVNFP uses up to 50% less NFV-nodes in comparison with distributed for provisioning SC requests (even though the blocking probability is almost the same). In other words, as activating NFV-nodes imposes additional costs, using DVNFP telecom operators are able to almost halve the SC provisioning costs. Finally in Fig 3 (c), it is interesting to note that, although DVNFP requires to activate less NFV nodes, it still provides lower violation of QoS (i.e., latency) requirements in comparison to the distributed case. This is due to the fact that DVNFP performs VNF grouping whenever latency requirements of a provisioned SC is not satisfied. Moreover, less latency violations can be observed with respect to the centralized scenario, as DVNFP is able to choose NFV-nodes based on the requirements of SC (i.e. nodes closer to the source for latency sensitive SCs are chosen).

VI. Conclusion

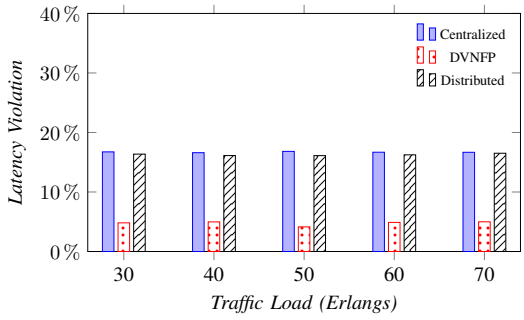
We proposed an algorithm for dynamic placement of VNFs in the network which reduces operation costs in metro by consolidating VNFs as much as possible in network nodes, while



(a) Blocking probability



(b) Average number of active NFV-nodes



(c) Latency violation

Fig. 3: Simulation results

maintaining a low blocking probability. Simulation results show that DVNFP algorithm can balance the trade-off among three different metrics (blocking probability, average number of active NFV-nodes and latency violation) outperforming completely centralized or distributed solutions. In this paper we considered a fixed value for the context switching delay. However, in future work we will provide additional analysis considering that context switching delay may vary according to the number of VNFs activated in a node, and to the amount of traffic processed by each VNF. As resiliency is one of the challenges that network operators face while deploying virtualized services (i.e, service chains) future steps will focus in extending this work to provide protection against link and or node failures for dynamic VNF placement based on reliability targets defined for each service.

VII. Acknowledgement

The work leading to these results has been supported by the European Community under grant agreement no. 761727 *Metro-Haul* project funding.

References

[1] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Dynamic chaining of virtual network functions in cloud-based edge networks," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*. IEEE, 2015, pp. 1–5.

[2] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-aware service provisioning in NFV-enabled enterprise datacenter networks," in *Network and Service Management (CNSM), 2016 12th International Conference on*. IEEE, 2016, pp. 153–159.

[3] 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*, 2017.

[4] J.-J. Pedreno-Manresa, P. S. Khodashenas, M. S. Siddiqui, and P. Pavon-Marino, "On the need of joint bandwidth and NFV resource orchestration: A realistic 5G access network use case," *IEEE Communications Letters*, vol. 22, no. 1, pp. 145–148, 2018.

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

[6] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint VNF placement and chaining," *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, pp. 19–24, 2016.

[7] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "Service chain (SC) mapping with multiple SC instances in a Wide Area Network," *arXiv preprint arXiv:1704.06716*, 2017.

[8] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1346–1354.

[9] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online VNF scaling in datacenters," in *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on*. IEEE, 2016, pp. 140–147.

[10] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Resource management and orchestration for a dynamic service chain steering model," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[11] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning," *IEEE International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.

[12] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. IEEE, 2015, pp. 191–197.

[13] F. M. David, J. C. Carlyle, and R. H. Campbell, "Context switch overheads for linux on ARM platforms," in *Proceedings of the 2007 workshop on Experimental computer science*. ACM, 2007, p. 3.

[14] V. Bobrovs, S. Spolitis, and G. Ivanovs, "Latency causes and reduction in optical metro networks," in *Optical Metro Networks and Short-Haul Systems VI*, vol. 9008. International Society for Optics and Photonics, 2014, p. 90080C.

[15] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE Press, 2012, p. 2.

[16] G. Xiong, P. Sun, Y. Hu, J. Lan, and K. Li, "An optimized deployment mechanism for virtual middleboxes in NFV-and SDN-Enabling Network." *TIIS*, vol. 10, no. 8, pp. 3474–3497, 2016.

[17] C. Westphal, "Challenges in networking to support augmented reality and virtual reality." ICNC, 2017.

[18] The Metro-Haul project deliverables. [Online]. Available: https://metro-haul.eu/deliverables/

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

[20] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. IEEE, 2015, pp. 191–197.

[21] A. Gupta. (2016) On service chaining using virtual network functions in operator networks. [Online]. Available: http://networks.cs.ucdavis.edu/presentation2016/Gupta-07-29-2016.pdf