Latency-aware Traffic Grooming for Dynamic Service Chaining in Metro Networks

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

***Abstract*—Optical metro networks are currently evolving in response to the new requirements of emerging 5G services. Network Function Virtualization (NFV) is being leveraged as a platform to dynamically provision these services on top of Virtual Network Functions (VNFs), and central offices in metro areas are being upgraded to host processing units that can host the needed to provision services with stringent-latency and high-bandwidth requirements closer to users (i.e., edge computing). By concatenating these VNFs in a specific order and route traffic among them, operators generate a so-called "Service Chain"(SC). Considering the fact that, new 5G services have bandwidth requirements typically with sub-wavelength granularity, traffic grooming is required to achieve efficient network resources utilization. Since grooming affects the end-to-end latency of provisioned services, we investigate how to perform latency-aware traffic grooming, and we propose an algorithm for dynamic SC provisioning, that considers the latency requirements of each SC to decide about if grooming shall be allowed at intermediate network nodes. Our proposed algorithm tries to minimize the blocked bandwidth as well as number of nodes to host VNFs in the network (NFV-nodes) considering the nodes computational capacity, links bandwidth and end-to-end latency constraints. Results obtained from numerical evaluation show that, our algorithm is able to reduce the number of NFV-nodes up to 50%, while keeping amount of blocked bandwidth below a specific threshold.**

***Index Terms*—NFV, VNF, Dynamic Service Chaining, Latency-aware, metro network**

I. Introduction

Service chaining of Virtual Network Functions (VNFs) has recently attracted lot of attention in the context of metro network, as metro is the network segment where future low-latency 5G services such as, e.g., Augmented Reality [1] will reside. As a response to these low-latency requirements, multi-layer optical networks that are based on Optical Transport Networks (OTN) over Wavelength Division Multiplexing (WDM) are being deployed. As an example, China Mobile is currently using this two-layer architecture to support its first deployment of 5G metro aggregation networks [2].

This kind of optical networks are characterized by high-capacity and low-latency. In conjunction, network operators are rethinking the architecture of optical metro nodes, which are evolving towards the so-called Central Office Re-architected as a Datacenter (CORD) [3]. Such evolution will enhance the functionalities of metro nodes, which, besides performing traditional switching and traffic aggregation, will also be equipped with processing and storage capabilities.

In this study we observe that the required bandwidth of emerging low-latency services is typically below the wavelength capacities used in WDM based metro networks, therefore, traffic grooming needs to be used to allow multiple low-speed connections with sub-wavelength granularity to be transported over shared lightpaths. With grooming, network capacity utilization improves and it is possible to save on electronic multiplexing equipment, by doing optical bypass. However, as grooming needs electronic processing, it will affect the end-to-end latency of the provisioned services. Therefore, a trade-off between network capacity utilization and tolerable latency arises.

In this paper we propose an algorithm to dynamically perform VNF placement, by chaining VNFs sequentially in a specific order, and routing traffic among them. Since the network comprises two layers, a latency-aware traffic grooming approach in metro network is required. The problem of VNF placement for service chaining has been very well investigated (see the next section) under single-layer settings, however, to the best of our knowledge, no existing work has considered the impact of traffic grooming on the performance of service chaining in dynamic metro area networks.

The reminder of the paper is as follows. In Section II we provide an overview of related works. Section III explains the problem addressed in this paper. In Section V the proposed algorithm is described. In Section VI we provide the simulated numerical results obtained comparing our algorithm with two benchmark algorithms namely, fully centralized and fully distributed. Finally, we conclude the paper in Section VII.

II. Related work

The problem of VNF placement has attracted lot of research , and especially dynamic VNF placement, in which dynamic changes of service requests are considered, is gaining lot of attention recently [4]. Ref. [5] provides an Integer Linear Programming (ILP) model for dynamic service chaining with the objective of maximizing the profit of the service provider. Ref. [6], provides an ILP formulation for an online VNF forwarding graph (an ordered set of VNFs, not necessarily in the form of a chain) with the objective of minimizing reconfiguration and routing costs and satisfying Quality of Service (QoS) of all service requests. Optimal VNF placement is found by both placing new VNFs and migrating existing ones. In [7], authors propose a dynamic VNF placement algorithm in a network consisting of three levels (edge, regional and nationwide level). They consider two different types of VNFs, data plane VNFs with stringent latency requirements, and control plane VNFs that are more tolerant to delays.

Their proposed algorithm places data plane VNFs in main Central Offices (COs) (edge level) while control plane VNFs and multi-access edge computing applications are placed in core COs (regional level) and centralized cloud platform (nationwide level). Ref. [8] provides a spatio-temporal model for mobile services requested by users in a specific region and, based on this model, it presents an algorithm for dynamic VNF placement. The proposed algorithm for each service request chooses the best location for an edge cloud to host VNFs, which is in vicinity of more user devices and is utilized less. If the predicted location for a VNF is different from its current location, it will be migrated. Authors in [9] propose a method for dynamic VNF placement using concept of modularly varying goals to tackle the high computational complexity of VNF placement problem. The method is called evolved VNF placement and it changes the objective of the problem in small intervals during adaption phase of the algorithm. Authors in [10] propose a dynamic queue scheduling model for efficient resource allocation taking into consideration the stability of the network. Then considering that network is stable, they formulate VNF placement problem into two sub-problems; Service Function Chain (SFC) scheduling and SFC mapping. They propose two heuristic algorithms to solve these sub-problems, and using these algorithms, they provide a queue-aware dynamic VNF placement algorithm. However, in none of the existing works the multi-layer aspect of this problem has ever been considered, i.e., no traffic grooming algorithm which considers the latency requirement of requested Service Chain (SC), as in this paper, has been proposed.

III. Problem statement

We aim at solving the placement of VNFs and grooming, routing and wavelength assignment (GRWA) to provision dynamically arriving SC requests. The problem can be formally stated as follows. Given an optical metro-network topology in which, for each dynamically generated SC request, we decide the placement of VNFs and GRWA of SC traffic, considering SC latency requirement as well as link and node capacity constraints. Our objective is to maximize the provisioned bandwidth of SC requests in the network while minimizing the number of active "NFV-nodes". Active NFV-nodes are the nodes with computational capabilities with at least one running instance of a VNF.

In the OTN-WDM network layer, to provision SC requests, in addition to NFV-nodes, there are forwarding nodes that do not perform any processing. In this network, as it is shown in Fig. 1, we have Metro Core (MC) and Metro Core Backbone (MCB) COs that are connected together using *core links* and are considered as NFV-nodes. Also Metro Aggregation (MA) COs exist in this network, that are connected to each other or to the MCBs and MCs using *extension links*. SC requests are generated dynamically at MA nodes (source node of SC) without computational capacity, and depending on the type of SC and its latency and computational requirements, a destination node is chosen among NFV-nodes.

Each SC request is characterized by an end-to-end latency budget and a set of virtual nodes, namely, a source and a destination node and a set of nodes hosting an ordered list

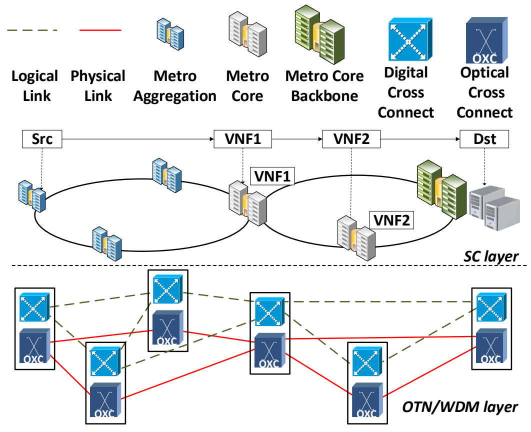


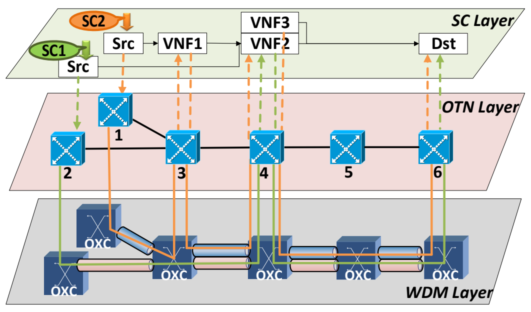
Fig. 1: Network model

of VNFs to be traversed by the SC. These virtual nodes are connected together using virtual links. As it is shown in Fig. 1, in order to provision a SC, its virtual nodes need to be mapped to physical nodes. Note that a VNF is deployed into a physical NFV-node using a certain number of virtual machines, each with a limited processing capacity, and leads to installation costs in terms of, e.g., software licenses and power consumption [11]. In addition, the virtual links connecting these virtual nodes, need to be mapped to a set of physical links which imposes link activation cost [12].

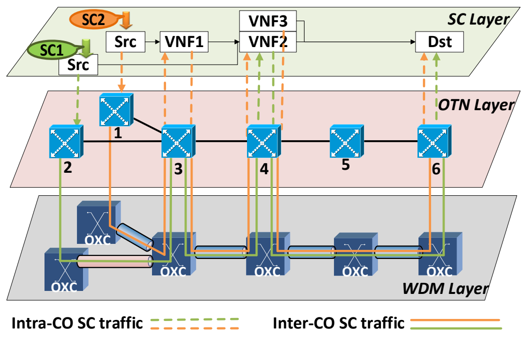
IV. Latency-aware traffic grooming

We categorize the SCs in three different latency classes based on their latency requirement, i.e., 1) *stringent*, 2) *intermediate* and 3) *loose*. For the scenario considered in this paper, stringent SCs have latency requirement in the order of 1ms, intermediate SCs can tolerate up to 5ms latency and loose SCs have latency requirement greater than 5ms.

Depending on the latency class of SCs, traffic grooming is applied less or more aggressively. In other words, for stringent-latency SC traffic, since grooming will impose additional latency on the SC due to electronic processing, traffic grooming is discouraged. However, for SCs with loose latency class, traffic grooming is more actively promoted. We call this approach "latency-aware traffic grooming". Fig. 2 depicts this approach through an example. We suppose that a SC supporting a Cloud Gaming service, which is composed by VNF2, with latency requirements larger or equal to 80ms, generated by users connected to node2, and destined for node6, is already provisioned in the network. As depicted in Fig. 2(a), a Smart Factory SC request that is composed by VNF1, VNF2 and VNF3, is generated at node1 and has node6 as the destination. This SC request supports a service with 1ms tolerated latency, so at node3 grooming is not done and we have two circuits destined for node6. However, upon arrival of VoIP SC request with 100ms latency tolerated by end-users, composed by VNF1, VNF2 and VNF3 and with node1 as source and node6 as destination, as it is shown in Fig. 2(b), traffic grooming is done at node3, since both SC requests use the same wavelength and they both belong to the loose latency class. Therefore, at node3 these low bandwidth circuits belonging to these two SC requests will be groomed into one high-capacity circuit.



(a) Traffic for two SCs is not groomed



(b) Traffic for two SCs is groomed at node3

Fig. 2: Latency-aware grooming

The latency model we considered in this paper is as follows. The end-to-end latency of the embedded SC (L_{e2e}) is calculated as the summation of propagation delay of each link l (τ_l) in the path computed for each SC request ($Path_{SC}$), and electrical processing delay at each node n, on the $Path_{SC}$ ($n \in Path_{SC}$), if traffic grooming is done at that node.

$$L_{e2e} = \sum\nolimits_{n \in Path_{SC}} T_{sw,n} + \sum\nolimits_{l \in Path_{SC}} \tau_l \qquad (1)$$

V. Dynamic VNF Placement and GRWA Algorithm

The dynamic VNF placement and GRWA algorithm (DVPG), that performs dynamic SC provisioning with latency-aware traffic grooming, is described in this section. In this algorithm, we leverage the following two topological features:

- *Betweenness centrality*: For each node, the betweenness centrality, quantifies the number of all possible shortest paths between any node pair, passing through this node.
- *Locality-awareness*: For each node, the locality-awareness value is defined as the length of the shortest path between source node of the SC and the node, summed up with the length of the shortest path between the node and destination node of the SC.

We detail the DVPG algorithm in the pseudocode in *Algorithm 1* and *Algorithm 2*. The DVPG algorithm takes as input the following set/parameters:

- *G(N,E)*: Current status of the network modeled as a layered graph
- F: Set of NFV-nodes in the network ($F \subseteq N$)
- V: Set of VNFs to be mapped in the network
- S: Types of SCs to be deployed
- W: Number of wavelengths for each link in the network
- r: SC request which is specified by these characteristics:
  - S_r: Source node of the SC request
  - D_r: Destination node of the SC request
  - $N_{vnf,r}$: Number of VNFs building the SC
  - V_r: The VNFs building this SC ($V_r \in V$)
  - L_r: Latency tolerated by users requesting SC
  - B_r: Bandwidth requirements of the SC
  - H_r: Holding time of SC request

As the first step, an auxiliary graph, according to the network topology considered, is built. In fact the auxiliary graph will be a layered graph with W+2 layers, in which, layers 1 to W correspond to W wavelengths while layer W+1 is the lightpath layer and the last layer is the access layer in which traffic flows are generated and terminated [13]. In this auxiliary graph we have different types of edges. At each layer W, there are *physical edges* that represent the physical links of the network. In the access layer the input and output ports can be connected to each other using *grooming edges* to represent traffic grooming capability at that node. An established lightpath between two nodes is represented by a *lightpath edge* and is composed of set of physical edges. Fig. 3 shows a simple example of an auxiliary graph. At each layer for each node, there are input and output ports. In this network we have three nodes and two links where each link supports n wavelengths and all the nodes have grooming capability. The blue arrow from output port of node1 to the input port of the node2 in W_1 layer, represents the physical link between node1 and node2. Upon establishing a lightpath between two nodes, an edge will be added to the lightpath layer of the graph, from output port of one node to the input port of the other node.

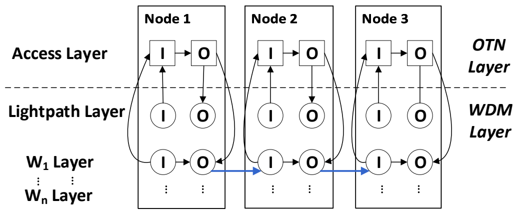


Fig. 3: Auxiliary graph

The SC requests arrive dynamically in the network. For a single SC request, our algorithm performs a set of steps which can be divided in two phases:

1) *Phase I*: In this phase, as it is shown in *Algorithm 1*, upon arrival of a SC request, depending on its latency class, different costs will be assigned to the graph edges. Note that in our algorithm, we use the hop count as the main cost metric. So, for stringent latency class, in the auxiliary graph, we have cost assignment that fosters utilization of direct paths in the network, hence for each physical link traversed by a lightpath edge, we assign 0.5 cost (to encourage using the residual capacity of already established lightpaths) and we assign cost equal to 0.6 (which is chosen to break ties between

similar solutions with and without grooming and discourage unnecessary grooming) to the grooming edge. For example, if we have a lightpath edge that traverses 2 physical links, it will have a cost equal to 1. For SCs in intermediate latency class, we use the same policy as stringent latency class for lightpath edges, but for the grooming edges we assign a very small cost. Finally, for latency loose SCs, the cost assignment for the grooming edges is the same as intermediate class, but for the physical links traversed by the lightpath edges we assign a cost equal to 1.1 which is slightly higher than the cost of a physical edge and will cause physical edges to be preferred over lightpaths, traversing the same number of physical edges. Note that for all the three latency classes, each physical edge has a cost equal to 1. In this way, latency-aware traffic grooming is performed. After that the first (next) VNF (V_r) belonging to the r is chosen to be mapped in the network. The main steps to perform dynamic VNF placement are discussed in our previous work [14].

2) *Phase II*: When all V_r are placed successfully on the NFV-nodes, the second phase of our algorithm starts. In this phase the L_{e2e} is calculated according to equation 1. Then, as it is shown in *Algorithm 2* line 2, the L_{e2e} is compared with L_r, If latency requirements of the SC is satisfied, the SC is provisioned and when its H_r is expired the bandwidth and computational resources used by this SC will be released. If L_{e2e} is higher than L_r, as it is depicted in *Algorithm 2* line 4, algorithm finds the virtual link in virtual path of SC ($VPath_{SC}$) with the highest amount of latency. Then algorithm tries to avoid using this virtual link by placing its VNFs on the NFV-nodes connected to the end points of this virtual link on $Path_{SC}$. After that, algorithm checks the L_{e2e} and will repeat the same procedure, until either the latency requirement of SC is satisfied or all the VNFs of the SC are consolidated on one NFV-node. If any of the above-mentioned steps fail, the counter for latency violated SCs will increase. At the end, algorithm checks whether there is a groomed lightpath in the $Path_{SC}$. If it is the case, the electrical processing delay will be added to the L_{e2e} of the existing SC requests, that were using this lightpath before traffic grooming was done. These steps are shown in *Algorithm 2* lines 5-25.

The DVPG algorithm uses Dijkstra algorithm to compute shortest paths on the auxiliary graph with $2 \times N \times (W + 2)$ vertices, hence, the complexity of it is $O(N^2W^2)$.

VI. Numerical results

To obtain numerical results we developed a discrete event-driven simulator in C++. We considered a reference metro-regional network of Telecom Italia similar to the topology shown in the Fig.4, with 52 nodes, out of which 2 nodes are MCB COs, 6 nodes are MC COs and rest are MA COs. In this network there are 72 bidirectional WDM links (10 core and 62 extension links). Core links connect MC nodes to each other and MCB nodes and are up to 200 km long. Extension links connect MA nodes to each other and MCs or MCBs. Each WDM link supports 8 wavelengths with 40 Gbit/s capacity. The SCs considered in this paper, their VNFs as well as their latency and bandwidth requirements are depicted in Table I while computational requirements of VNFs are shown in Table II. We consider that incoming SC requests are randomly selected among those shown in Table I, following a uniform

Algorithm 1: Placement of Virtual Network Functions

```
1: Given:network state G(N,E) and set of NFV-nodes $F$, Set of VNFs $V$, Service Chain request $r(S_r, D_r, N_{vnf,r}, V_r, L_r, B_r, H_r)$
2: Build auxiliary graph
3: repeat
4:     Select the $r$
5:     Update cost of graph's edge base on latency class of SC
6:     repeat
7:         Select the next $V_r$
8:         if $\exists$ $F_{vnf}$ already in the network then
9:             Select all $F_{vnf}$ with enough capacity.
10:            Sort all $F_{vnf}$ by increasing value of $length(shortestpath(S_r, F_{vnf}))$
11:            if $\exists$ more than one $F_{vnf}$ with equal $length(shortstpath(S_r, F_{vnf}))$ then
12:                Sort all $f \in F_{vnf}$ and select only $f$ that satisfy: $loc_f - length(shortestpath(S_r, D_r)) < \delta$.
13:                if $\exists$ more than one such $f$ then
14:                    Choose $f$ with less $Num(V_{act})$.
15:                    if $\exists$ more than one $f$ with the min $Num(V_{act})$ then
16:                        if SC requires computational capacity then
17:                            Choose the $f$ closer to MCs.
18:                        else
19:                            Choose the $f$ closer to Src.
20:                        end if
21:                    end if
22:                end if
23:            end if
24:            Try to scale up the $V_r$ on $f$ until success or all $F_{vnf}$ have been tried.
25:            if success then
26:                update G(N,E)
27:                continue
28:            end if
29:        else
30:            Select in order the $F$ on $shortestpath(Src, Dst)$.
31:            Try placing the $V_r$ on an $f \in F$ until all $f \in F$ on $shortestpath(S_r, D_r)$ have been tried.
32:            if failed then
33:                Select the $f$ with less $loc_f$
34:                if $\exists$ more than one $f$ with $min(loc_f)$ then
35:                    select $f$ with higher betweenness centrality.
36:                end if
37:                Try placing the $V_r$ on a $f$ until all $f$ have been tried.
38:                if failed then
39:                    return $r$ blocked
40:                else
41:                    Update G(N,E)
42:                end if
43:            else
44:                Update G(N,E)
45:            end if
46:        end if
47:    until all $V_r$ are chained
48: until all $r$ are provisioned
```

Algorithm 2: SC Latency Improvement

```
1:  Calculate $L_{e2e}$ of the embedded SC
2:  if $L_{e2e} > L_r$ then
3:      repeat
4:          Select the virtual link with highest latency.
5:          Release the resources of the VNFs on its end-points.
6:          Find the two closest nodes to two end-points on $VPath_{SC}$ with enough capacity.
7:          if Such node not found then
8:              Increase counter for latency violated $rs$
9:              return SC request provisioned
10:         else
11:             Enable those VNFs on these two nodes
12:             Replace virtual link between those two nodes with virtual link in line 4 in SC path.
13:         end if
14:         if Consolidated all virtual links and latency not satisfied then
15:             Increase counter for latency violated $rs$
16:             return SC request provisioned
17:         end if
18:     until Latency satisfied or all VNFs have been consolidated
19: end if
20: if $\exists$ a groomed lightpath in SC path then
21:     Update $L_r$ of SCs using that groomed lightpath before grooming
22: end if
23: Provision SC request.
24: Release the resources when $H_r$ expires.
25: return SC request provisioned
```

distribution. The considered VNFs are Network Address Translation (NAT), Firewall (FW), Video Optimizer (VO), Traffic Monitor (TM), Intrusion Detection System (IDS). SC requests are generated at MA nodes dynamically. In this network there are 14 NFV-nodes each equipped with 512 CPU cores. All the nodes in the network have grooming capabilities, and wavelength conversion is enabled only in case of grooming of a connection. The SC requests are generated with an inter-arrival rate λ that follows a Poisson distribution and a holding time according to negative-exponential distribution, with mean holding time μ=1 second. All the results are obtained with a confidence level of 95% with at most 5% confidence interval on blocking probability.

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

Service Chain	Service Chain VNFs	Bandwidth	Latency
Augmented Reality	NAT-FW-TM-VO-IDS	100 Mbps	1 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	NAT	FW	VO	TM	IDS
CPU Core	0.0184	0.018	0.108	0.266	0.214

A. Benchmark algorithms

To evaluate the performance of our algorithm, we considered two benchmark algorithms:

- *Centralized*: In this algorithm, the main objective is to have a lower bound on the number of active NFV-nodes, by using just one NFV-node with unlimited computational capacity located at the MCB, to serve all the SCs. In this case, high SC blocking probability will be obtained, due to the high network congestion experienced in the links adjacent to the NFV-node.
- *Distributed*: In this algorithm, the main objective is to reduce SC blocking probability by activating all VNF instances in all NFV-nodes. Each SC request is served using the closest NFV-node to the user, by activating all the VNFs that it needs on the NFV-node. Blocking probability is minimized, but all the metro edge nodes must be active all the time.

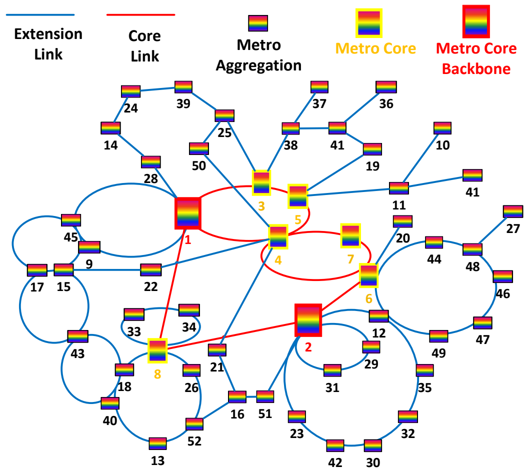
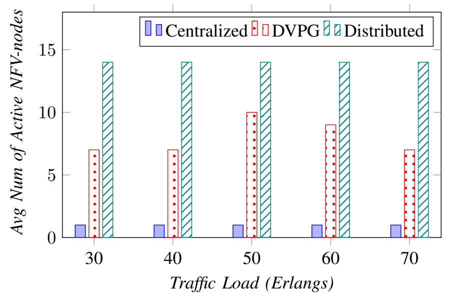


Fig. 4: Network topology

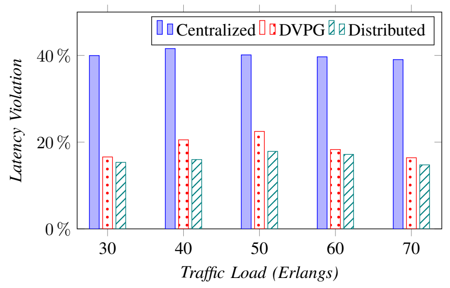
B. Performance Comparison

In this subsection we discuss the obtained numerical results. We conduct the experiment for different traffic loads and consider three different metrics to evaluate the performance of our algorithm with respect to the benchmark algorithms: *i) average number of active NFV-nodes*, that is calculated as number of active NFV-nodes weighted by the amount of time each NFV-node is serving SC requests; *ii) latency violation ratio*, calculated considering latency-violated provisioned SC requests out of the total number of SC requests; *iii) bandwidth blocking*, calculated as the percentage of blocked SC requests' bandwidth out of total bandwidth of all SC requests.

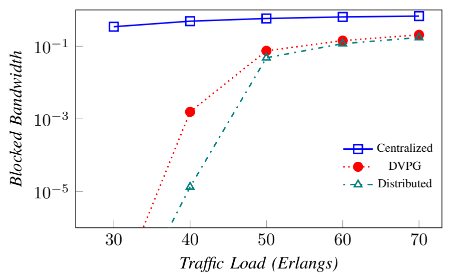
As it is depicted in Fig 5(a), for different traffic loads, average number of active NFV-nodes for DVPG, always lies between the values for *Centralized* and *Distributed*. Note that, DVPG activates up to 50% less NFV-nodes with respect to *Distributed*, as it tries to reuse VNFs in the network as much as possible. In other words, as the first step to place a VNF, DVPG searches for already active VNF instances in the network. If no active VNF instance is found, or existing VNF instances in the network can not be scaled up, a new VNF instance will be instantiated on an NFV-node. We notice that for DVPG, for traffic loads up to 50 erlangs the number of NFV-nodes increases with traffic load. However, after that, we can see that there is a downward trend for average number of NFV-nodes. It is due to the fact that, for higher loads, the links are more congested, so algorithm tends to activate new NFV-nodes and consolidate all the VNFs of each new SC on one NFV-node, typically dedicated to the provisioned SC. As for



(a) Average number of active NFV-nodes



(b) Latency violation



(c) Blocked bandwidth

Fig. 5: Simulation results

the latency violation ratio, that is shown in Fig 5(b), DVPG maintains a value close to the *Distributed*. It is due to the fact that after choosing the NFV-nodes to place the VNFs of the SC request on them, if the latency requirement of the SC is not satisfied, DVPG tries to consolidate VNFs as much as possible to minimize the latency violation ratio (as described in Section V, *Algorithm 2*). Since *Distributed*, for each SC request, using the shortest path, chooses the closest NFV-node to the user, it has the lowest latency violation ratio. As it can be inferred from the Fig 5(b), for high traffic loads, for all the three algorithms, the latency violation ratio decreases with traffic load. The reason is that, when the load in the network is high, there will be higher blocking probability, which will result in less number of provisioned SCs. Hence, as latency violation is just checked for provisioned SCs, there will be less latency violated SCs.

As it is illustrated in Fig 5(c), DVPG, maintains a value for the blocked bandwidth in between *Distributed* and *Centralized* approaches. However, for higher traffic loads, DVPG can perform almost as good as *Distributed* which represents the lower bound for blocked bandwidth because it always chooses the closest NVF-node to the user to provision a SC request.

VII. Conclusion

In this paper we proposed an online and scalable algorithm to dynamically solve problem of latency-aware traffic grooming for service chaining in metro networks. By performing latency-aware traffic grooming, our algorithm is able to utilize bandwidth resources efficiently, whenever the latency requirement of SCs allows to do so. We conducted a number of simulations using a discrete-event driven simulator. Results show that our algorithm is able to balance the trade-off among different metrics namely latency violation ratio, average number of NFV-nodes and blocking probability. Our algorithm enables network operators to save up to 50% of service provisioning cost by activating less number of NFV-nodes while maintaining an acceptable blocking probability and latency violation ratio. As to perform traffic grooming an auxiliary graph is used, the time complexity of DVPG is relatively high. As for the future step, we will focus on providing an algorithm with lower computational complexity.

Acknowledgment

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

References

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

[2] C. Zhang. (2018) "Optical Networking in the Cloud and 5G Era," Keynote talk at Optical Fiber Communications Conference and Exhibition (OFC). [Online]. Available: https://www.ofcconference.org/en-us/home/ news-and-press/ofc-video-library

[3] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 96–101, 2016.

[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] 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.

[6] N. T. Jahromi, S. Kianpisheh, and R. H. Glitho, "Online VNF placement and chaining for value-added services in content delivery networks," *arXiv preprint arXiv:1806.04580*, 2018.

[7] F. Slim, F. Guillemin, A. Gravey, and Y. Hadjadj-Aoul, "Towards a dynamic adaptive placement of virtual network functions under ONAP," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017 IEEE Conference on*. IEEE, 2017, pp. 210–215.

[8] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[9] M. Otokura, K. Leibnitz, Y. Koizumi, D. Kominami, T. Shimokawa, and M. Murata, "Application of evolutionary mechanism to dynamic virtual network function placement," in *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–6.

[10] L. Tang, H. Yang, R. Ma, L. Hu, W. Wang, and Q. Chen, "Queue-aware dynamic placement of virtual network functions in 5G access network," *IEEE Access*, vol. 6, pp. 44 291–44 305, 2018.

[11] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 343–356, 2017.

[12] A. Leivadeas, M. Falkner, I. Lambadaris, G. Kesidis, C.-H. Lung, and M. Ibnkahla, "Considerations for a successful network service chain deployment," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 288–292.

[13] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 285–299, 2003.

[14] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *2018 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE, 2018, pp. 136–141.