QoS Impacts of Slice Traffic Limitation

Khalil Mebarkia, and Zoltán Zsóka

***Abstract*—Slicing is an essential building block of 5G networks and beyond. Different slices mean sets of traffic demands with different requirements, which need to be served over separated or shared network resources. Various service chaining methods applied to support slicing lead to different network load patterns, impacting the QoS experienced by the traffic. In this paper, we analyze QoS properties applying a theoretical model. We also suggest appropriate parameter setting policies in slice-aware service function chaining (SFC) algorithms to increase QoS. We evaluate several metrics in different analysis scenarios to show the advantages of the slice-aware approach.**

***Index Terms*—Network Slicing, QoS, SFC, VNF, 5G Networks**

I. Introduction

Emerging communication technologies like 5G allow the provision of services with extended requirements. For example, new application sets can be served, which might combine high data rates, low latency, and extended reliability needs to be satisfied by the network.

Besides the progress in the radio networking part, which allows higher access and data rate for the clients, the control and data plane handling in the core part include innovative solutions. One of these is the support of Virtual Network Functions(VNFs), a technique for distributing the elaboration steps of traffic among some nodes instead of loading only central ones. VNF-capable nodes allow to start/scale/stop elaboration functions in virtual machines realized through various virtualization techniques.

The building blocks and management architecture of VNF-based solutions are described by a standard of the European Telecommunications Standards Institute (ETSI) [1]. Typical functions to be virtualized are Firewall, balancing, compression, and shaping of the traffic load. If a series of VNFs have to be considered in the provision of a service request, the task of Service Function Chaining (SFC) has to be performed to select appropriate VNF-capable nodes. Then, the traffic should pass through this chain of serving nodes to get the required elaborating functions.

A further novel concept introduced in 5G is *slicing*. It allows the definition of multiple service sets and a set of networking or even infrastructure resources to serve their requests. Fig. 1 illustrate this concept.

Since slices represent different types of traffic with different statistical properties and quality requirements, the requests belonging to them need appropriate handling with special SFC and routing solutions. The service chains apply both functional and networking resources as VNF-capable nodes and transport

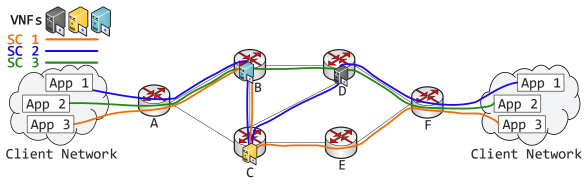

Fig. 1. Slice Concept of different services in different slices

links, respectively. At the same time, the network-related part of Quality of Service (QoS) needs can be satisfied only with settings in the networking devices.

The task of providing slices is twofold. On the one hand, higher-level problems as request admission, VNF resource selection, orchestration, or pricing and billing require intelligent control plane functions. On the other hand, we have problems for the lower level, like assignment of VNF and network resources and adjust settings in devices for handling QoS requirements of slices.

In this paper, we analyze slice-aware SFC mechanisms from the network QoS point of view. We propose different policies for adjusting the SFC parameters with the queue-level settings and evaluate their behavior. We consider only the VNF functional capability of the nodes and neglect their exact resource limits. Our concept concentrates on using network resources, and we aim to preserve them for other slices to hold the QoS expectations.

The paper is structured as follows. Section II summarizes the results of related works. In Section III we present methods allowing QoS in slicing, and we define the most important analysis metrics. In Section IV we propose policies for parameter setting in QoS-aware Service Chaining algorithms. We analyze the policies in Section V We conclude the paper in Section VI.

II. Related works

Papers [2], [3] address NFV as a promising architecture proposed to increase the scalability and the functionality of the network by leveraging virtualization technologies. It describes how telecommunication networks and services are designed and operated when traditional Network Functions (NFs) get transformed into VNFs.

Different approaches have been proposed by industry players, research institutes, and mobile operators to standardize SFC. In [1], ETSI defines a network service as a chain of VNFs. It emphasizes the demand for a new set of orchestration and management functions.

ETSI defines SDN usage in an NFV as an architectural framework and proposes a framework with three main components: VNFs and two subsystems termed respectively Manage-

K. Mebarkia and Z. Zsóka are with the Department of Networked Systems and Services, Budapest University of Technology and Economics, 1117 Budapest, Magyar tudósok körútja 2, Hungary (e-mail: mebarkia@hit.bme.hu; zsoka@hit.bme.hu).

DOI: 10.36244/ICJ.2021.3.3

ment and Orchestration (MANO) and Network Function Virtualization Infrastructure (NFVI) where VNFs are deployed. While in the RFC 7665 [4], IETF defines the service function chain as an ordered set of abstract service functions and ordering constraints. IETF also describes an SDN based SFC architecture. In this, an SFC classifier in the data plane performs a classification of end-to-end traffic to determine which VNF should be chained to process the traffic based on its requirements.

Various scientific works address the placement and chaining of VNFs. The proposed solutions focus on selecting the proper nodes for deploying a VNF for a specific traffic demand, or solve the SFC problem assuming the network topology and the demands, with VNF capabilities and VNF requirements, respectively. For instance, the authors in [5] propose a placement algorithm, which takes into consideration hardware accelerator resources in addition to compute resources. They aim to optimize the use of resources in Network Function Virtualization Infrastructure (NFVI), while placement algorithms must consider the presence of accelerators in NFVI nodes. They describe an Integer Linear Programming (ILP) for the accelerator-aware VNF placement problem. In [6], the authors study the VNF placement problem in SDN/NFV-enabled networks. They formulate the problem as a Binary Integer Programming (BIP) in which they aim to minimize a weighted cost, including the VNF placement cost. The authors propose a Double Deep Q Network-based VNF Placement Algorithm (DDQN-VNFPA) using deep reinforcement learning.

Other papers address similar problems while considering multiple slices in the network. Although Network Slicing is one of the most crucial parts of 5G core networks, its definition has never been unique, clear, and precise. It is varying from different perspectives of the various service providers. For instance, Next Generation Mobile Networks (NGMN) [7] defines a network slice as a set of network services that consists of 3 layers, Service Instance Layer, Network Slice Instance Layer, and Resource Layer. The network slice runs on top of physical resources where network services and resources conform to a logical network to deliver specific requirements. Slice can also be defined as a set of network and VNF resources, which can support one or more services, each with a prescribed series of VNFs that the service traffic shall pass. The supported services can be told to *be in the slice*.

The authors of [8] formulate the problem of statically embedding service chains into slices while also considering network link capacities. In [9] a Mixed Integer Linear Program (MILP) formulation is given for the problem of optimizing slices over multiple domains and accepting multiple services in each slice. The authors also present a heuristic that can guarantee the QoS requirements for the services by allocating the needed resources for the slices. Another work on the optimal allocation of VNF resources in 5G networks with cross-domain slices is [10], which introduces an ILP formulation and a Multi-layer based Knap-sack-based heuristic. Their solution aims to minimize the number of VNFs hosting the functions that constitute different network slices. Various QoS metrics are taken into account while the slices' set gets reorganized each time a service request arrives.

The authors in [11] present models for sliced networks, and investigate the cost reduction promises of using the NFV and network slicing technologies. In the models the slice deployment costs are allocated to show the network efficiency with slicing, while considering one specific demand that is realized as a service consisting of chained VNFs.

The authors in [12] focus on the end-to-end life-cycle management of network slices on different sites using a single management and orchestration entity with a coherent proof of concept. They propose algorithms for efficiently activating, deactivating, and decommissioning the network slices, using real-time status information from Network Slice Management Function (NSMF). The results show that by adopting a better strategy in these algorithms for controlling various phases of the slice life-cycle, the response time can be reduced for a user request by 50%.

Paper [13] presents a survey of works on slice admission control, citing and grouping works of various methodology. It presents multiple objectives for admission control, from revenue optimization to fairness, and mentions the priority-based strategy. Note that instead of admission priorities, our paper speaks about packet service priorities in the network queues, which is a different aspect.

The authors of [14] consider slices with demands with uncertainty in their number and requested resources. Their model involves a probabilistic approach of provisioning the node and link resources to fulfill the requirements. The problem of mapping the uncertain demands on the resources is formulated as a nonlinear constrained optimization problem, and then it is reduced to a parameterized a mixed integer linear programming (MILP) problems. The consideration of the uncertainty allows mappings that might be used in dynamic scenarios too.

Paper [15] extends the slice demand mapping problem to include also guarantees on the end-to-end latency of the traffic and to use a combined objective for the optimization. The authors consider the option of flexible routing, or in other words, load balancing of traffic via multiple paths, and present a mixed binary linear program (MBLP) formulation for the problem. In their model, the latency calculation considers only the propagation delay and a static NFV delay while neglecting the queueing delays at the network nodes. Due to the high complexity of the full formulation of the problem, a reduced formulation is contributed too. The slice mapping solution presented in [16] also applies multiple paths, but for a different reason. The paper takes under the scope another important requirement for SFC, and design slices with guaranteed availability.

In [17], the service chaining problem is considered in a two-layer model, which consists of a Functional Layer (FL)and a Network Layer (NL). We logically separate the topology of VNF-capable nodes and functional links allowed among them and the topology of network nodes and links. We address the problem of considering the current load state of both the functional links and the network below it. We discuss how to determine the SC according to the required bandwidth and VNF order while avoiding overloads on the network links. As a result, we propose heuristic and ILP solutions to formulate these challenges. These solutions are based on the dynamic calculation of SC by considering the current network load to avoid the use of heavily loaded links. The heuristic algorithm *OdAASP* (Overload Avoiding Augmented Shortest Path) determines the shortest path between source and destination with the awareness of considering overload avoidance. As a comparative solution, we use the algorithm *SFC-CSP* (SFC-Constrained Shortest Path) that finds the shortest path and satisfies a given SFC constraint as proposed in [18].

In [19], we introduce heuristic service chaining solutions that consider shared slicing and apply a kind of preservation of network resources for other slices to hold the QoS expectations. The application of these solutions can control the network link loads in several situations. Our objective now is to discuss systematically the concept of slicing-awareness by resource preservation, and to extend the analysis to more detailed QoS properties. Our aim is to show the importance of handle network slicing considering shared resources and dynamic service requests, to ensure the low latency and guaranteed bandwidth for different services. Our experience is that this topic is not well discussed in the state-of-the-art works.

III. QoS in Slicing

Slices are assumed to be service sets allowing multiple requests and using a determined set of network and VNF resources. [8] defines the sharing property for VNFs. This value describes how the available VNF resources can be shared among slices. Since we concentrate on the network resources, the model is extended to links and simplified to the two basic cases: fully shared resources and lack of sharing. We consider the network as a two-layered system:

FL the Functional Layer contains logical connections, which connect traffic end-nodes and VNF-capable nodes. It implements the chains of VNFs.

NL the Network Layer contains the IP connections, which connect traffic end-nodes, VNF-capable nodes, and networking nodes. It implements the network paths.

sharing among slices can be considered then in NL only or both layers. We assume the latter case and full sharing.

A. Slicing model concept

From the service requirements point of view, in our simplified model concept, each slice defines a traffic type with an ordered set of required VNFs and QoS values. Moreover, this traffic type is described with the high- or low-level traffic parameters, as request arrival rate, interarrival time, and length of packets. For example, let us refer to a slice supporting voice and another supporting real-time video calls.

Dedicating the resources to slices helps to provide guarantees, but can lead to suboptimal usage and lower throughput in several situations when dynamic traffic changes occur. The analysis of this concept is out of our current focus. We assume no dedication in the shared model, i.e., all the slices can use any network resources. A mechanism for coordinating the use of resources in FL and NL is needed to support the requirements. How the VNF resources like CPU/time or memory can be assigned to traffic requests of different slices is out of our scope now, and we assume no limitation in the functional layer.

In the networking layer, we can assume the resource sharing supported by traffic management techniques like in DiffServ or IntServ model of IP. To follow our simple concept of slicing, the class-based approach of DiffServ can be enough for handling slice traffic on IP links. In this case, the link capacity sharing can be implemented with weight-based queueing like Weighted Fair Queueing (WFQ) or its version Low Latency Queueing (LLQ), which also supports strict priority class.

This time we consider only unicast requests. It is worth mentioning that not like in many other works, e.g., in [11], in our concept, the slice is not restricted to only one possible pair of end-nodes. Instead, it supports a set of such relations, i.e., traffic requests of the same slice can have different endpoints. The important is that they are of the same type.

Various technologies can solve the implementation of the two-layer model and the slice traffic management in NL:

- GRE (or other) tunnels can realize FL links,
- IP routing, like OSPF can realize the mapping of FL links to NL links,
- MPLS-TE, or IPv6 can provide traffic classification at the entry nodes and class-based handling on links.

B. Modelling Queueing and Overloads

WFQ and LLQ are weight-based serving policies for queueing, which allows the share of link capacity among the traffic of different classes. The class load, or in our case, the slice traffic, can be adjusted on the links in many ways. Let us refer here to two basic cases.

In traffic engineering, weights are set on each link separately, according to the relation of admitted traffic load coming from the supported classes. On the other hand, in the case where the class preferences are determined preliminary and independent from link loads, the required relation of classes can be *coded* in the weights. For instance, we can say that in general, the traffic of slice S_1 shall get twice as large bandwidth as slice S_2. Then we can set the same weights on every link according to the preliminarily determined values. Our model considers this second approach, and no strict priority class is used now.

One can calculate QoS properties for a queueing system using theoretical models, primarily based on the Markovian approach or its extensions like Markov arrival processes (MAP), or quasi-birth-death processes (QBD). These models are stable when the relative load is less than 1, i.e., there is no overload on the system. However, we have to study also networks where the traffic dynamics can lead even to link overload situations. In such cases, the effective load of slice traffic gets reduced to the part that can pass through the link, because the part over the link capacity gets thrown with high probability.

To catch this behavior, we use a simplified approach instead of the exact stochastic model. The model considers directed traffic loads and link capacities, and we illustrate it in Fig.

2. The figure presents three different load use-cases for a link shared among the *red*, *violet* and *green* slices with WFQ weights of $w_r = 0.2$, $w_v = 0.3$ and $w_g = 0.5$. The dashed line shows the link capacity, and the fourth column illustrates the high load case showing also the parts thrown away from the *red* and *green* slice traffic due to overload.

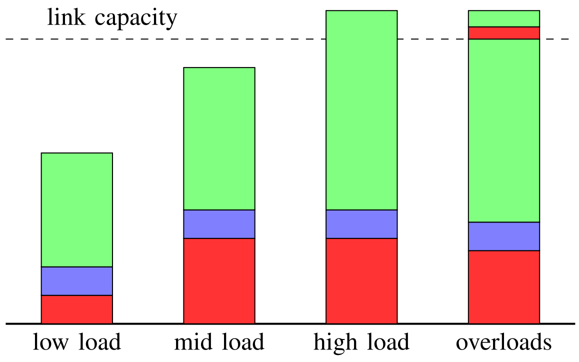

Fig. 2. Link capacity sharing in different load cases

Let B_i be the requested (or offered) bandwidth of slice/class S_i demands on a link, while the average experienced bandwidth for this traffic be $\overline{EB}_i$. Note that on each link l with capacity C_l, we have:

$$C_l \geq \sum_{S_i \in \mathrm{S}} \overline{EB}_i,$$

and,

$$B_i \geq \overline{EB}_i, \forall S_i \in \mathrm{S}.$$

In the simplified model, we assume that in the case of a lower or middle load of a link, i.e., without overloads, the weights do not play a significant role in the level of averages. Thus, we assume:

$$\overline{EB}_i = B_i, \forall S_i$$

For an overloaded link with capacity C_l and using weights w_i in WFQ, we have two cases. Let $\mathrm{S}^{ul} \subset \mathrm{S}$ the subset of slices requesting *less bandwidth than possible*, i.e., where:

$$B_i \leq w_i C_l$$

The model calculates the average bandwidths as follows:

$$\forall S_i \in \mathrm{S}^{ul} : \overline{EB}_i = B_i \quad (1)$$

$$\forall S_i \notin \mathrm{S}^{ul} : \overline{EB}_i = \left(C_l - \sum_{S_u \in \mathrm{S}^{ul}} B_u \right) \frac{w_i}{1 - \sum_{S_u \in \mathrm{S}^{ul}} w_u} \quad (2)$$

The average bandwidth of slices in subset S^{ul} is easily calculated. For the other slices, we start from the capacity remaining for them, and share it according to the WFQ weights normalized on this subset of slices.

The simple model can be extended to consider a strictly prioritized slice S_p too. For S_p, we have:

$$\overline{EB}_p = min(B_p, C_l) \quad (3)$$

The capacity C_l of the link has to be decreased by $\overline{EB}_p$ before the subset S^{ul} gets selected, and the further calculation is performed.

We might use this simple model easily for a single link, but we are in a much more complicated situation with a network of links or queues. As best, we should handle this case by a reduced-load approximation, an iteration on the requested and average bandwidths of slices. However, in this work, we use a simplified approach also for this issue.

C. *QoS metrics*

The simple model might determine average bandwidth values even for overloaded situations, but a QoS analysis requires a more accurate approach, like that proposed in [20]. It might provide packet-based waiting time and packet loss probability values, and show their dependence on slice weights. We propose a combination of these two levels to get metrics for our analysis.

First, a macroscopic model is involved in handling overloads. According to the simple model above, for each slice, the required and experienced bandwidth will be lost on an overloaded link. Therefore, its interpretation can be a slice load reduction, which comes from overloading. To describe the factor of reduction on a link, we define the value:

$$OvlRed_i = \frac{B_i - \overline{EB}_i}{B_i}$$

A higher factor means more fraction of lost traffic.

To avoid instability, we apply the stochastic model with input parameters mimicking a reduction by $OvlRed_i$ factors, i.e., the analysis can be done for traffic not overloading the link. The simplest way is to enlarge the mean of interarrival time, although higher moments might be affected too. Thus, the waiting times and packet loss results are valid for the part of the traffic that is not thrown due to overloading.

Note that the *macroscopic* loss $OvlRed$ is very important and might be greater than the *microscopic* packet loss by magnitudes if the system is overloaded. Therefore, the macroscopic loss also needs to be considered when comparing the waiting times of different mechanisms or network loads.

From the network point of view, link-level values, i.e., the means and higher moments of the mentioned metrics on single network links might be important. However, for us, more important are the QoS values regarding the traffic requests. Therefore, we extend the proposed metrics starting from the link-level values to end-to-end values as in [21].

We calculate the mean end-to-end latency and packet loss probability metrics for the traffic request r^i coming from slice S_i by applying the simple forms:

$$E^{tr}_{r^i}(W) = \sum_{l \in P_{r^i}} E(W_{l,i}) \quad (4)$$

$$p^{tr}_{r^i} = 1 - \prod_{l \in P_{r^i}} (1 - p_{l,i}) \quad (5)$$

Set P_{r^i} is the set of network links used in the whole service chain assigned to the request.

For the $OvlRed$ metric, we use a different approach than for the microscopic loss because the reduction on one link l of the path P_{r^i} strongly impacts the traffic that arrives at the following link. Therefore, the reduction values can not be considered as independent ones. To model this, we take the maximum value on the path, i.e., we calculate:

$$OvlRed_{r^i} = \max_{l \in P_{r^i}} (OvlRed_{l,i}) \tag{6}$$

For the sake of simplicity, we neglect the accurate reduced-load approximation for both microscopic and macroscopic metrics. Since for each slice requests the service chains need to meet an ordered series of VNFs, the path in the network layer NL can even contain loops, and it is hard to consider them in the iteration.

IV. Slicing-aware Service Chaining

A. Service Chaining Algorithms

Most of the algorithms proposed for service chaining handle traffic requests independently, considering only the required resources like bandwidth or VNFs. Some of them might concentrate only on the number of used networking resources as SFC-CSP [18] does, which ignores the overloading effect. Other algorithms like OdAASP in [17] try to avoid the overloaded network resources if it is possible, but still not consider that the sharing of a resource is usually done on a class or slice basis. It can lead to inefficient use of the links and higher traffic loss when the load increases.

In [19] we present solutions that involve the concept of slicing without systematic introduction of whole class of the algorithms. Their main advantage is the resource preservation for the future requests of a slice. SLF and SLN algorithms use the limitation of resource usage and can control the loads coming from the slices on the Functional or Networking Layer links, respectively. The applied mechanism is simple: the link cost gets vastly increased before the path selection if a special limit rate SL for the given slice would be overridden on the link by leading the chain or the network path of the current request through it.

Both algorithms are based on the lowest cost chain solution SFC-SP [18]. That finds the lowest cost chain from the source s to the destination d of the service request r_i of slice S_i and bandwidth bw_{r_i} considering the series of VNFs prescribed for r_i.

In *SLF*, first of all, the cost $c(l^F)$ of the functional link l^F gets modified to a relatively high value like 10^6 times greater than its original cost, if:

$$bw_{r_i} + \sum_{r \in \mathrm{R}(S_i, l^F)} bw_r > SL(S_i, l^F) \times B(l^F) \tag{7}$$

$\mathrm{R}(S_i, l^F)$ is the set of the already chained requests that are from slice S_i and contain link l^F in their chains. In our two-layer network model, the capacity $B(l^F)$ is calculated as the bottleneck capacity on the network link path to that l^F is mapped.

After modifying costs, the lowest cost chaining finds a chain excluding the links where the slice traffic would be over the limit. Note that the Slice Limitation concept could be applied with any other service chaining algorithm.

The algorithm *SLN* works very similarly to *SLF*, but the relative load limitation is taken into account on the network links of NL. Since more than one functional link can be mapped on a network link, their load cannot be considered independent. The QoS-based prioritization is done on the Network Layer's resources; thus, from *SLN* we can expect service chaining that is more adjusted to packet serving.

The load limitation with *SLN* is based on the values $SL(S, l^N)$ set for each slice S and network link l^N. The algorithm *SLN* starts with the modification of the cost $c(l^F)$ of each functional link l^F, where the mapping of l^F contains at least one network link l^N with

$$bw_{r_i} + \sum_{r \in \mathrm{R}(S_i, l^N)} bw_r > SL(S_i, l^N) \times B(l^N) \tag{8}$$

The functional links that violate the limitation get a high cost, leading to the use of network links, which are not so much loaded by the slice S_i.

On all links, we set the limit value for each slice. The results show that these solutions allow resource sharing among slices according to a preference system defining the priorities or weights of slices. Such kind of preference system is realized as the WFQ or LLQ systems on networking resources. On the other hand, in the low and middle range of load, the algorithms behave nicely also from the overloading point of view since they balance the load among the resources.

B. Slice limitation and QoS

The performance of the algorithms SLF and SLN depend strongly on the limiting parameter SL_i. There are two basic cases of setting SL_i for slice S_i:

- in the simple case, we set the same values on each link *uniformly*, e.g., according to the preferences regarding the slices,
- in the generic case we can set *different* values on every link, e.g., according to the estimated slice load on the link.

Although the generic case might perform better when the network behavior is well estimated or the weighting and limiting parameters are often adjusted, dynamic slice request changes can lead to unstable situations in such a case.

To avoid it, we consider the same settings for queueing weights on each link, and we use the simple setting case in this work. However, it remains a question, which value shall be set as limit SL_i. We might precisely adjust the limits to the set of WFQ weights realized in queues and reflect the provider preferences on the slices, but this is not the only way. We aim to provide and compare different setting policies for the slice-aware algorithms.

C. Adjusting policies

We propose three policies for adjusting the limitation parameters. For a clearer view, we assume normalized w_i weights in WFQ, i.e., $\sum_{S_i \in \mathrm{S}} w_i = 1$. The limit parameters can be set as it follows:

Conservative (Cons) The value:

$$SL_i = \min_{j \in \mathrm{S}} w_j$$

on each link, i.e., the bandwidth of each slice is intended to be pushed below the load of the least weighted traffic class. It means that SFC starts to use low loaded links quite early. This policy is supposed to work nicely only for the case when the weights are similar.

Weight-aware (WeiAw) The value:

$$SL_i = w_i$$

on each link, i.e., the links shall be loaded with slice traffics according to their weights. This policy should preserve as much bandwidth resources for a slice, as it would take with the queueing.

Liberal (Lib) The value:

$$SL_i = \max_{j \in \mathrm{S}} w_j$$

on each link, i.e., the limit for a slice can be higher than its weights. It means that SFC starts to use low loaded links quite late. This should work well in cases where the traffic request pattern is not aligning well with the queueing weights.

Enabling service chains with maybe more hops but less loaded links shall affect the QoS metrics. On the one hand, the higher number of hops in routes can enlarge the end-to-end delays and lead to more queues where the traffic might suffer packet losses. However, on the other hand, we can expect lower values in link-level results for delays and losses in the case of moderate network load.

When the network load elevates strongly, any of these simple policies can lead to situations with high costs on many functional or network links. Thus, the SFC tends to use the shortest chain for many traffic requests, and link overloads can appear.

V. Evaluation of the Proposed Methods

We compare the above introduced QoS results for the policies in topologies of different scales. We have implemented the proposed solutions in the framework presented in [22]. The SFC is performed in this simulation tool considering the order of the requests arrival, but not allowing any departures. Based on the link-loads experienced in the simulator and considering the assumed QoS characteristics of slices, we calculate the metrics applying the theoretical models of [20] and the extensions proposed above.

A. Small topology

First, let us introduce the analysis using the network topology illustrated in Fig. 3, which also shows the VNF capabilities in nodes. Network links are of 1Gbps capacity, and one-to-one mapping is applied between FL and NL. We assume two slices, traffic requests of slice S_0 (red) and S_1 (blue)require VNFs v_0 and v_1 respectively. In the experiment,

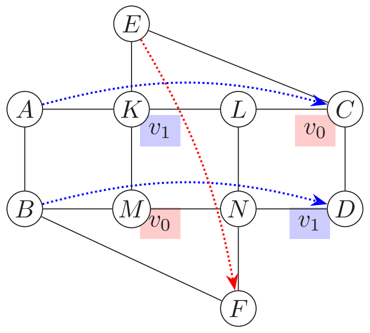

Fig. 3. Toplogy with service chains

we evaluate a four-phase elevation of slice traffic, adding more and more requests to slice S_0 between the node pair $E-F$. The traffic of S_1 is less increasing, and its requests are between node pairs $A-C$ and $B-D$. The number of requests to chain, the average of the demanded bandwidth, the mean packet lengths, and the assigned WFQ weights are summarized for each slice in Table I. We apply the WFQ weights on each link uniformly.

TABLE I
TRAFFIC TYPES CHARACTERISTICS

Traffic	Number	Bandwidth	Packet Length	Weight
Slice S_0	1-8	0.3 $Mbps$	8 $Kbit$	0.6
Slice S_1	2-3	0.35 $Mbps$	500 $Kbit$	0.4

Fig. 4 presents on its two y-axes the average values of macroscopic and microscopic loss calculated for the S_1 traffic requests. The policies proposed for the limitation-based algorithm SLN are compared with each other and the simple SFC-CSP. In this case, we applied a one-to-one mapping for links in FL and NL; thus, SLF and SLN are identical.

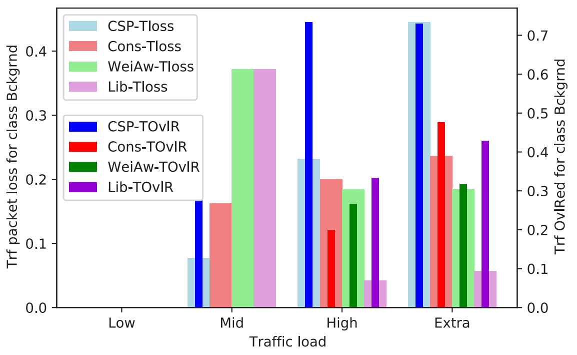

Fig. 4. Average traffic-level loss for slice S_1

As expected, in the case of relatively low and mid-range traffic load, all the metrics are moderate. The elevation of the load induces the elevation of microscopic packet loss, except for the SFC-CSP algorithm, where the overload of links appears already. The best-performing policy here is the conservative one.

In the higher load ranges, we can see that due to the overload of network links, the overload reduction plays the primary role in the loss, while microscopic packet loss decreases. The macroscopic reduction of traffic ends with less loaded links and thus with lower packet loss values. In SFC-CSP, the chains are static, and in SLN with the liberal policy, we accept higher slice loads on links before we start to use alternate chain paths. This behavior leads to higher overloads.

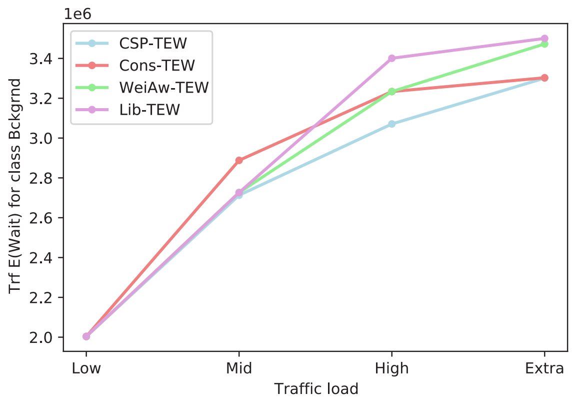

Fig. 5. Average traffic-level waiting time for slice S_1

In the case of SLN with weight-aware, and especially with conservative policy, the traffic suffers from a different effect. The higher load induces a pretty early use of alternate chain paths and balance the load among more links, but these chains might be of more hops. Having more hops in NL means more links where the traffic and the packets can get lost. On the other hand, when the network load gets high, nearly all network resources can get overloaded because traffic flows everywhere. The results show that considering both macroscopic and microscopic loss we get the lowest values for the conservative policy if the load is not too high. For extra high load cases, the best performing policy is the weight-aware, where the SL_i limits are adjusted to the weights used in WFQ.

The above explanations clarify the results on the end-to-end delay of S_1 requests presented in Fig. 5. Although suffers from significantly higher losses, algorithm SFC-CSP, with its relatively short chain paths, performs well from this perspective. The SLN goes better with conservative and weight-aware policies, while the liberal one performs poorly.

B. Larger Topology

The network under the scope is the hypothetical backbone network of Algeria, which is used in [17] too. Fig. 6 presents the topology of the IP layer, which contains 10 Core Routers (black-filled nodes) and 17 Edge Routers (grey-filled nodes) each at different sites.

We assume only one type of IP connection of 10Gbps capacity, attached to Core Routers and Edge Routers. In addition, 27 eNodeBs are distributed on the 27 sites, and three different VNFs are placed by random placement resulting in the following *Core* nodes:

- v_1 is placed in Algiers and Boussada,
- v_2 is placed in Algiers and Tenes,

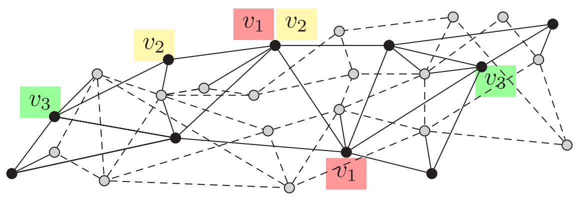

Fig. 6. IP layer topology and VNF placement

- v_3 is placed in Oran and Constantine,

The links of the functional layer FL form a full graph between these three nodes extended by those connecting the eNodeBs to these nodes. The mapping onto network links in NL simply uses the shortest paths.

There are two simplex traffic demand types, in other words, slices, to be served, New Services and Best Effort, referred to as S_0 and S_1, respectively. Each traffic demand requires to pass through the VNF-series v_3, v_2, v_1. The demands of the New Services type start from one eNodeB in Algiers to all eNodeBs. Each traffic demand of type Best Effort starts and ends in randomly chosen eNodeBs. We aim to study a scenario where the type of New Services S_0 bandwidth grows linearly from 0 up to $1500 Mbps$. The traffic demands arrive in a randomized order. The numbers of demands, the average of the demanded bandwidth (in Mbps), packet length, and WFQ weights are summarized for each traffic type in Table. II. In the SL-based algorithms we apply the Weight-Aware policy.

TABLE II
Traffic Types Characteristics

Traffic	Number	Bandwidth	PacketLen	Weight
New Serv. (S_0)	27	$0 - 1500$	$8\ Kbit$	0.6
Best Effort (S_1)	702	$3, 98$	$500\ Kbit$	0.4

Fig. 7 and 8 present the average end-to-end packet delay calculated for slices S_0 and S_1.

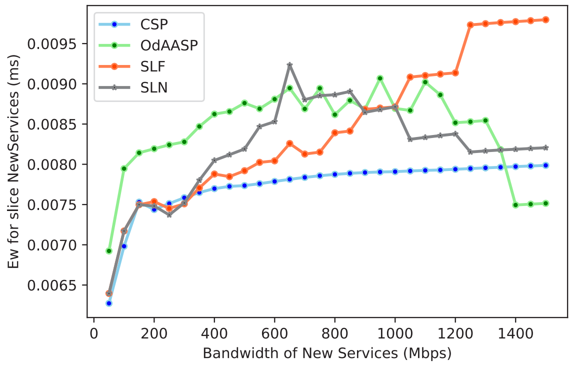

Fig. 7. Average traffic-level delay for slice S_0

In the low-load range, below 450 Mbps, CSP and SL-based algorithms perform nearly the same while values for OdAASP are slightly higher. In the mid-load range, 450-900 Mbps, on the one hand, the SL-based methods show higher waiting time, which comes from finding low-loaded links in the corresponding slice and use them. The network resources get exhausted in the high-load range, over 900 Mbps, where we observe very high waiting times. Here we also face a descending trend in OdAASP and SLN, which comes from returning to choose often SFCs with the links of the shortest network paths.

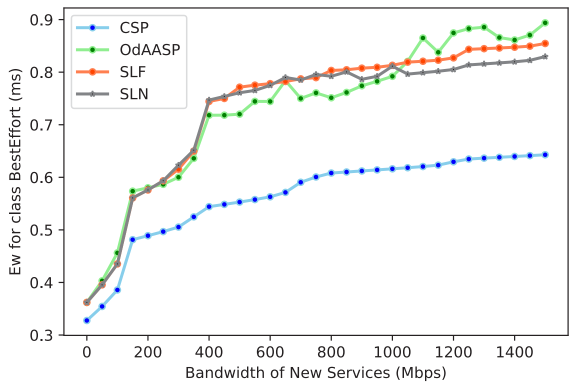

Fig. 8. Average traffic-level delay for slice S_1

We observe similar behavior in Fig. 8 for the other slice, where the SLN algorithms perform in the same way as OdAASP, without the descending trend at high loads. In the case of CSP, the waiting time values are low because chains' paths contain few links in both the Functional and Networking Layer.

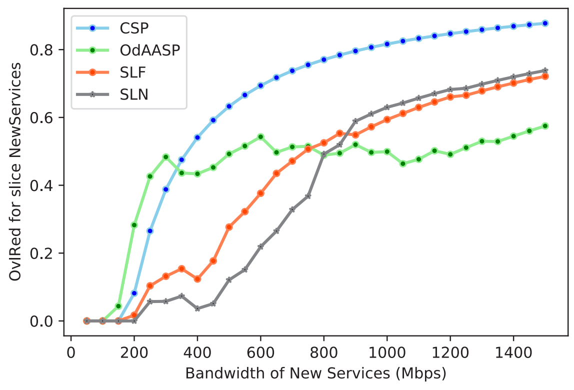

Fig. 9. Average traffic overload reduction for slice S_0

In the CSP case, it is evident that the waiting time is lower than for the others since it always uses the shortest paths when chaining the demands. The other algorithms do this when the load is high, and there are already many demands using the links in FL or NL. However, as expectable, always using the same links in CSP leads to overloads even for moderated network load. In Fig. 9 we observe that the demands of S_0 suffer a pretty high overload reduction with CSP. The overload avoiding algorithm OdAASP performs worse than the SL-based ones in the low or middle load phase due to the high number of demands in slice S_1. Unlike SLF and SLN, OdAASP does not spare resources for S_0 traffic and might chain it over short but overloaded paths.

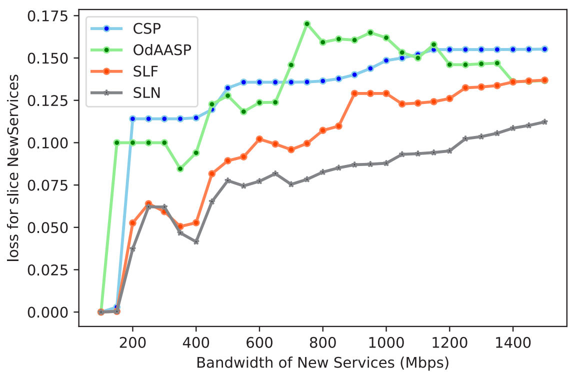

Fig. 10. Average traffic-level loss for slice S_0

Fig. 10 presents the end-to-end packet loss or microscopic loss that happens on links after throwing away the overloading parts. Also, here we observe that the SL-based algorithms perform better than CSP and OdAASP, although the difference is not that large for higher loads. The cause of the fall-back effect at about $300Mbps$ is that link overloads appear in the network. The macroscopic traffic reduction affects the microscopic metrics in a good direction, i.e., lower losses and waiting times.

We observe that SLN with weight-aware policy goes pretty better than any other algorithm. As the relative load limitation is considered on the network links of NL, it takes more chains that can be mapped on a network link. The QoS-based prioritization is done on the Network Layer's resources; thus, from SLN, we can expect service chaining that is more adjusted to packet serving.

VI. Conclusion and Further Work

This paper focuses on SFC methods, which support slices, and consider their packet level handling during the calculation of the appropriate service chain. It proposes different policies for setting up the parameters of the SFC methods. The model behind the methods ignores the load and latency details or limitations of VNFs, but considers link capacities and network loads coming from the different slices, which share the available resources according to the implemented queueing. This allows the systematic evaluation of QoS properties that can be experienced on the links or by the service requests. Result values are calculated with a packet-level model of network links extended by a macroscopic loss concept that shall handle overloads. The concept is not strictly coupled to the NFV, and might be applicable for architectures based on containerized functions.

The calculation model could be extended to analyze algorithms' performance for slices supporting time-critical applications. Besides the average latency, we could calculate their maximum values or at least the probability of overriding a given delay threshold.

The numerical results show that the proposed algorithms perform better from several QoS metrics point of view than those missing the slice-aware property. From a set of results seems that the conservative policy works well when the load is moderate, while for higher loads the weight-aware policy can perform better. Another result-set demonstrates the advantages of method SLN in the middle load range, although the evaluation is not straightforward due to the complex dependencies among the different kinds of measures.

Moreover, handling the requests dynamically while optimizing the SFC separately for the slices may lead to handle the slices in an unfair manner. As a next step, the further analysis from fairness point of view shall be done. The extension of the proposed policies might help to catch this issue.

References

[1] ETSI. Network functions virtualisation (nfv)management and orchestration. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf (Accessed 2014-12)).

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb 2015.

[3] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[4] J. Halpern and C. Pignataro. (2020) Service function chaining (sfc) architecture. [Online]. Available: https://datatracker.ietf.org/doc/rfc7665/?include_text=1

[5] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, "Vnf-aap: Accelerator-aware virtual network function placement," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2019, pp. 1–4. doi: 10.1109/NFV-SDN47374.2019.9040061

[6] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou,"Optimalvnfplacementvia deep reinforcement learning in sdn/nfv-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2020.

[7] N. alliance. 2016 ngmn 5g p1 requirements and architecture work stream end-to-end architecture description of network slicing concept. [Online]. Available: https://www.ngmn.org/wp-content/uploads/160113_NGMN_Network_Slicing_v1_0.pdf (Accessed 2014-07-15).

[8] T. Truong-Huu, P. Murali Mohan, and M. Gurusamy,"Servicechainembedding for diversified 5g slices with virtual network function sharing," *IEEE Communications Letters*, vol. 23, no. 5, pp. 826–829, May 2019.

[9] R. Addad, M. Bagaa, T. Taleb, D. L. Cadette Dutra, and H. Flinck, "Optimization model for cross-domain network slices in 5g networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[10] R. A. Addad, M. Bagaa, T. Taleb, D. L. C. Dutra, and H. Flinck, "Optimization model for cross-domain network slices in 5g networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1156–1169, 2020.

[11] A. Chiha, M. V. D. Wee, D. Colle, and S. Verbrugge, "Network slicing cost allocation model," *Journal of Network and Systems Management*, vol. 28, no. 3, p. 627–659, 2020.

[12] S. Vittal, M. K. Singh, and A. Antony Franklin, "Adaptive network slic- ing with multi-site deployment in 5g core networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 227–231. doi: 10.1109/Net-Soft48620.2020.9165512

[13] M. O. Ojijo and O. E. Falowo, "A survey on slice admission control strategies and optimization schemes in 5g network," *IEEE Access*, vol. 8, pp. 14 977–14 990, 2020.

[14] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Uncertainty-aware resource provisioning for network slicing," *IEEE Transactions on Network and Service Management,* vol. 18, no. 1, pp. 79–93, 2021.

[15] W.-K. Chen, Y.-F. Liu, A. De Domenico, and Z.-Q. Luo, "Network slicing for service-oriented networks with flexible routing and guaranteed e2e latency," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5. doi: 10.1109/SPAWC48557.2020.9154330

[16] R. Gour, G. Ishigaki, J. Kong, and J. P. Jue, "Availability-guaranteed slice composition for service function chains in 5g transport networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 13, no. 3, pp. 14–24, 2021..

[17] K. Mebarkia and Z. Zsoka, "Service traffic engineering: Avoiding link overloads in service chains," *Journal of Communications and Networks*, vol. 21, no. 1, pp. 69–80, Feb 2019.

[18] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications,* 2018, pp. 2132–2140. doi: 10.1109/INFOCOM.2018.8485996

[19] Z. Zsoka and K. Mebarkia, "Slice-aware service chaining," in *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2021, pp. 6–12. doi: 10.1109/ICIN51074.2021.9385550

[20] A. Horvath, G. Horvath, and M. Telek, "A joint moments based analysis of networks of map/map/1 queues," *2008 Fifth International Conference on Quantitative Evaluation of Systems*, pp. 759–778, 2008.

[21] K. Mebarkia and Z. Zsoka, "Qos modeling and analysis in 5g backhaul networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6. doi: 10.1109/PIMRC.2018.8580739

[22] B. Farkas and Z. Zsoka, "Augmenting sdn by a multi-layer network model," in *2016 European Conference on Networks and Communications (EuCNC)*, 2016, pp. 215–219. doi: 10.1109/EuCNC.2016.7561035

Khalil Mebarkia received the M.Sc. degree in Computer Science (Networks and Multimedia) from University of Bordj Bou Arreridj, Algeria in 2016. He is currently working toward the Ph.D. degree in Computer Engineering at Department of Networked Systems and Services in Faculty of Electrical Engineering and Informatics of Budapest University of Technology and Economics, Hungary. His research interests include Computer Networks and Protocols, Software Defined Networks, Virtualized Network Function, and Cloud Computing.

Zoltán Zsóka received his M.Sc. degree in 1999 in Technical Informatics from Technical University Budapest, Hungary and his Ph.D. degree in 2007 from the same University. He is currently associate professor in Department of Networked Systems and Services, in Faculty of Electrical Engineering and Informatics of Budapest University of Technology and Economics. His research interest includes Networking Virtualization and Automatization, and Software Defined Networks.