Real-time Maintenance of Latency-sensitive 5G Services through Network Slicing

Rafael Montero*, Fernando Agraz, Albert Pagès, Salvatore Spadaro
Universitat Politècnica de Catalunya (UPC)
Jordi Girona 1-3, 08034, Barcelona, Spain
*e-mail: rafael.montero@tsc.upc.edu

Abstract—Network Slicing appears as one of the enabling technologies for 5G networks to accommodate services with different requirements and availability. The present work, seizes the use of network slicing and focuses on the deployment and maintenance of services which are sensitive to latency constraints. For this purpose, the design and use of a VNF Latency-sensor are presented, considering its aggregation to the internal chain of services and the retrieval of latency data from the sensor. In this way, and by making use of such data, the expected service performance can be guaranteed. A multi-level orchestration and control architecture is then introduced to provide all the required functionalities for this mechanism. In order to assess this method experimentally, an emulated multi-segment testbed, considering specific 5G network segments (i.e., Access, Metro and Core), is used. The experimental results demonstrate the correct latency sensing of a particular slice and the process of service maintenance through the triggering of proper network actuations such as path reconfiguration or slice reallocation.

Keywords—5G, Network Slicing, Slice Composition, Service Chaining, Latency Sensor, Optical Networks, Actuations.

I. INTRODUCTION

The current trend towards highly programmable, highly reliable networks for the upcoming 5G services has brought new challenges to current network scenarios. In this regard, a set of defined service types for 5G [1], comes along with very demanding requirements (e.g., high throughput, low latency, etc.), which becomes a crucial matter for service operators. In turn, operators now need to accommodate current implementations to handle these new types of service. To this end, they must provide the required resources for each of them from a commonly shared infrastructure, besides establishing a required level of isolation among services so their expected performance can be maintained in time.

Network Slicing has been introduced as the enabling technology to leverage legacy networks towards all these new requirements. The slicing concept entails partitioning (i.e., slicing) either physically or virtually network resources so these can be shared among services with different requirements, without affecting each other. Each slice can allocate one to many services related to a tenant, so multiple tenants can coexist over the same infrastructure. To achieve such coordination, a Management and Orchestration (MANO) entity at a high level of the architecture [2] must be in charge of managing slices, as well as of coordinating configuration messages towards lower-level management components (e.g., controllers, orchestrators) across the network. In this way, the correct provisioning and maintenance of slices and services can be guaranteed.

Other technologies such as Software Defined Networking (SDN) [3] and Network Functions Virtualization (NFV) also come into play to facilitate slice configuration and operation. SDN in turn, enables networks to become fully programmable by separating the control and data layers, then logically centralizing control tasks to open the path towards open source network programming. In this way, networks dependence on vendor specific hardware is avoided. As a result, SDN allows abstracting and exposing underlying network infrastructure resources towards upper layers in order to optimize their utilization. In the case of NFV, the use of Virtual Network Functions (VNFs) is introduced, which enables the virtualization of specific functionalities to bring more flexibility to networking.

VNFs in this regard, could represent functions allocated all over the network infrastructure across many different segments (e.g., Access, Metro and Core). As a service could be composed of one to many VNFs, it is required that these are connected in a specific order. This fits to the concept of Service Chaining, which consists on interconnecting or chaining a set of functions to enable a particular service operation. Handling these interconnections besides managing the provisioning and maintenance of VNFs, services, slices, requires high-layer coordination between different management components and must be considered in any end-to-end 5G network scenario [4].

A. **Related Work**

Taking into account the slice lifecycle management and all the types of services defined for 5G, it is necessary to focus on the different requirements set for the proper operation of each of them. Among these requirements, latency is considered as one of the most demanding and crucial, in both deployment and maintenance stages of a slice. In this regard, at the provisioning stage, there already exist works focused on the use of latency data to properly handle service deployment [5] [6]. Besides providing an architecture capable of handling all the required network coordination, there were presented certain ways to use up-to-date latency data at service provisioning by introducing high-level service chain computation elements or by proposing the use of orchestration algorithms.

Regarding the maintenance stage of a slice, the need is set in sensing specific parameters (e.g., latency) that could provide data about the current state of running services, and being able to use such data to act over the network. Such actions (i.e., actuations), done either preventively or predictively, intend to make the required changes in a way to keep guaranteed the correct service operation. In this sense, works such as in [7], present real-time system control of networks, providing as well an architecture and ways to demonstrate network reconfiguration. However, there is still a need to apply the concepts of real-time service monitoring and maintenance to the scope of network slicing for 5G.

With our work presented in [8], we covered latency awareness at the provisioning stage considering the deployment of slices along with the use of an introduced latency sensor. Moreover, we presented an architecture capable of orchestrating all required actions and a testbed to experimentally prove the sensor provisioning. This process, entailed the deployment of the mechanism used to measure the current latency at a 5G service, based on the comparison of time stamps at analysed packets. To this end, network elements require of an accurate clock synchronization following the use of protocols such as Network Time Protocol (NTP) or Precision Time Protocol (PTP) [9]. Finally, we introduced the collection of measured latency data for its processing at higher-level management components.

B. **Contributions**

In this paper, we follow up the work presented in [8], to put the focus on the real-time maintenance of slices/services, considering latency constraints. In this sense, we emphasize on the gathering of latency data from a running service and how this data is used to validate agreed service policies. We demonstrate this by proving the latency sensing experimentally. Furthermore, we analyse cases for actuations driven by the violation of established latency performance levels. In particular, we exemplify cases for path reconfiguration and slice reallocation [10], providing workflows to describe the interactions required between management components from highest to lowest level of the architecture.

The work following this introduction is structured as detailed next. Section II describes the overall maintenance architecture from a sensing and actuation point of views. Then, Section III characterizes the latency sensing mechanism and its deployment in form of a VNF sensor. Section IV details the experimental results of this work, considering latency sensing measurements and use cases for actuation scenarios. Finally, Section V concludes this work summarizing its achievements.

II. ARCHITECTURE FOR SERVICE MAINTENANCE

As discussed in the introduction section, managing the maintenance of a Network Slice (NS) relates to the high-level coordination between a set of management components in charge of control/orchestration functions. In this sense, an architecture tailored for this purpose is presented in this section considering software and hardware elements at different levels. The defined management components in turn, provide the necessary functionalities to handle the sensing and actuation process required to achieve slice maintenance.

The architecture is shown in Fig. 1, representing network elements and management components distributed from lowest (bottom) to highest (top) level in a multi-segment 5G scenario. Starting at the physical layer, segments of the network such as Data Centres (DC), Metro/access and Core are depicted, where some consider the allocation of both network and computing/storage resources (e.g., DCs). In this case, SDN controllers provide network control while network orchestrators running on top, also known as Virtual Infrastructure Managers (VIMs), manage the whole segment virtualized infrastructure. As for segments allocating only networking resources (e.g., Metro, Core), these could be driven by one to many WAN SDN controllers. Controllers and orchestrators in turn, present characteristics particular to each segment, which relate to the management of specific networking technologies (e.g., electrical, optical) and types of resources (e.g., physical, virtual).

A SDN controller then, is able to expose towards higher layers, all the information and capabilities of underlying physical network resources. In addition, it provides the tools to configure and reconfigure the network, thus allowing it to change according to real-time service needs. The orchestrators in turn, in segments such as DCs, communicate directly with controllers to provide them of the required network configurations, so these get pushed to network elements. If there is no orchestrator at the segment, this data could also be provided by higher-level entities. Besides this, orchestrators are also in charge of managing the allocation and instantiation of virtual computing resources (e.g., VM, VNF, container).

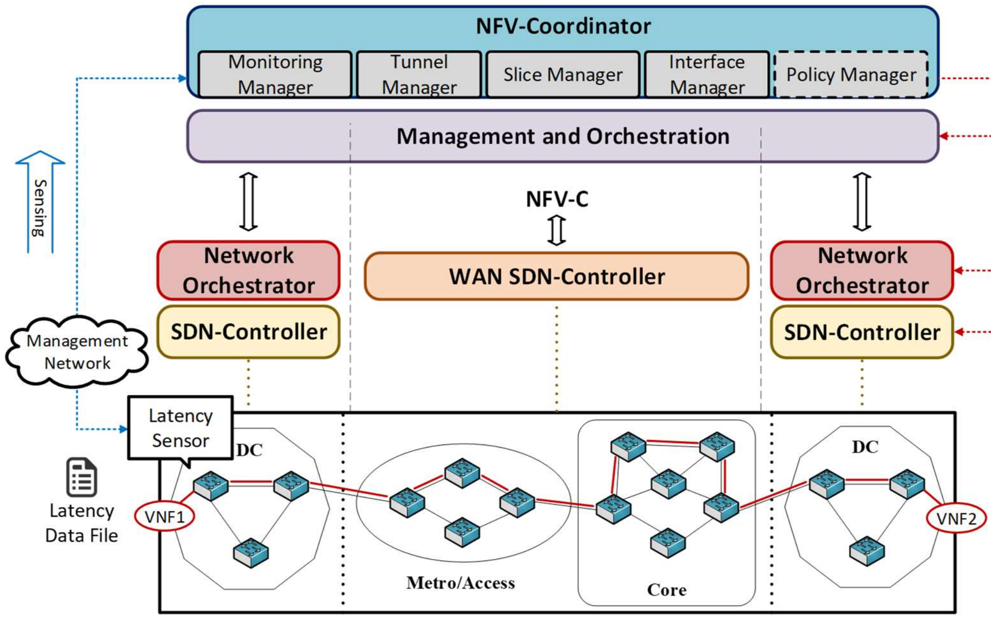


Fig. 1. Multi-level Multi-segment Slice Maintenance Architecture.

At a higher level resides the MANO entity, which coordinates the slice deployment across all registered VIMs. Moreover, it manages the description files (i.e., blueprints) that provide the data regarding the types of resources/configurations required for VNF and NS creation, which can also be used to modify the structure of a deployed slice. In turn, these become crucial when high-level reconfigurations are required (e.g., slice reallocation).

On top of the whole architecture, we introduce the NFV Coordinator (NFV-C) module, to act as the middle point between the network and external clients, such as Operation Support Systems (OSS) / Business Support Systems (BSS) or 5G Verticals. In this way, the NFV-C provides not only the required messaging between all set of network segments for slice deployment, but also the monitoring of real-time parameters to become aware of each slice current performance levels. Next subsections, characterize the internal modules of the NFV-C used to gather all this data, besides describing how it is used to detect degradations in the service quality standards (i.e., considering client-agreed service performance levels), and how preventive actions are executed over the network in response.

A. **Monitoring**

At the NFV-C and during the provisioning stage, the Slice Manager (SM) module, which in turn is the one coordinating overall slice deployment steps, instructs the Monitoring Manager (MM) module with the specific parameters to gather at each particular slice. More specifically, the parameters associated to the Key Performance Indicators (KPI) and the expected Quality of Service (QoS) levels agreed with the service client. Once the slice is provisioned and the service is running, the MM starts gathering these parameters from network elements and management components at different levels of the architecture. The collected data is then stored at the MM database, classifying it per-monitored datapath, which entails the end-to-end connectivity route between two VNFs at the slice level. In this way, awareness on the current state of running services and configured datapaths is kept.

In case of latency monitoring, which is the focus of this work, the data is gathered directly from a VNF latency sensor via an external management network. An in-depth look on how latency is retrieved from running services is given in Section III. In respect of other types of parameters (e.g., BER, throughput, VM CPU/RAM consumption), the MM could contact other sources such as SDN controllers, WAN controllers, Orchestrators, etc.

After data reaches the MM level, the specific parameter information is sent to the Policy Manager (PM) module. The PM also receives at the provisioning stage from the SM, the established thresholds for each parameter that is being monitored. In this way, the PM is able to check if the data retrieved from the running service fits the expected performance levels. If defined thresholds are not satisfied, then the actuation functions are triggered.

B. **Actuation**

Performing actuations over the network means triggering reconfigurations to adapt the slice to the current state of the network with the purpose of maintaining a desired level of service performance. In this regard, the PM is the module that takes such decisions upon violations of established policies. In terms of guaranteeing latency, two particular cases are analysed in this work, where each requires different actions to take place as explained next:

- ***Path Reconfiguration:*** If either a network congestion or failure worsens the latency performance of an existing service, the PM uses the data stored at the MM database to compute a candidate path, to continue ensuring the required service levels. Once the new path is selected, the PM triggers the path reconfiguration process via the Tunnel Manager (TM) module at the NFV-C. The TM at this point, contacts segment SDN/WAN controller(s) to send reconfiguration instructions. Once these are pushed to the corresponding network elements, the new datapath becomes operative and the monitoring process start again to check if latency has gone back to safe levels.

- ***Slice Reallocation:*** In case a network congestion due to the high amount of traffic flowing to multiple VNFs allocated at specific section is identified, the PM checks via the SM if there are available locations to perform a VNF migration, in a way to lower the current traffic bottleneck. Upon an affirmative answer from the SM, the PM consults the MM database, and calculates the best available path to reach the new VNF destination considering the current state of configured datapaths. Once the PM has all the requisites, it triggers the slice reallocation process by sending requests to the SM, TM and Interface Manager (IM) modules, asking for VNF migration, path reconfiguration and routing reconfigurations respectively. The SM then sends slice modification instructions to the MANO entity, which in turn pushes these via the correspondent VIM orchestrator. The TM in turn, instructs the WAN controller(s) to configure the selected datapath. Finally, the IM pushes new routing configurations to the VIM related SDN controllers to complete the migration. At this point, the slice is reallocated and traffic congestion is expected to decrease. To check this, the MM starts collecting again latency data to validate policy compliance at the PM.

III. LATENCY SENSING

After analysing how the MM and PM use the latency data to guarantee service performance by considering policies and actuations, it is necessary to describe the way this information is retrieved from the data layer. In this section, we present the sensing mechanism used to measure latency from a service currently deployed and running, besides introducing the VNF-based latency sensor and its role at the maintenance stage.

A. **Sensing Mechanism**

With the goal of measuring latency, a sensor is placed in the form of a new VNF instance in between two VNFs associated to a particular service (i.e., at the service chain). The sensor contains the mechanism used to measure latency and a local database to store this data. More specifically, the sensor can be configured to analyse any type of TCP data traffic flowing through, by setting the port associated to the data flows whose latency has to be monitored. The idea resides on the sensors capability to use packets time delay, given by the difference in the time stamp for each packet and its corresponding ACK, to calculate the round-trip time (RTT) between VNFs and thus the latency. Detailed steps are given next:

- *Step 1:* The sensor uses the time stamp of the first packet coming from VNF1 to get the Packet-LEFT arrival time (T_{PL}).
- *Step 2:* When the ACK of this packet is received by the sensor from VNF2 it uses as well its time stamp to get the Packet-LEFT-ACK arrival time (T_{PLA}).
- *Step 3:* On the other side the same process begins, with the first packet coming from VNF2 the sensor gets the Packet-RIGHT arrival time (T_{PR}).
- *Step 4:* An ACK from VNF1 is also received in response to this packet, then getting the Packet-LEFT-RIGHT arrival time (T_{PRA}).
- *Step 5:* The sensor uses T_{PL} and T_{PLA} to get the round-trip time between the sensor and VNF2 (*RTT-R*), it also calculates *RTT-L* on the other side by using T_{PR} and T_{PRA}.
- *Step 6:* Having calculated all this side to side *RTT* information between VNF1 and VNF2, it is possible for the sensor to finally get the latency between them.

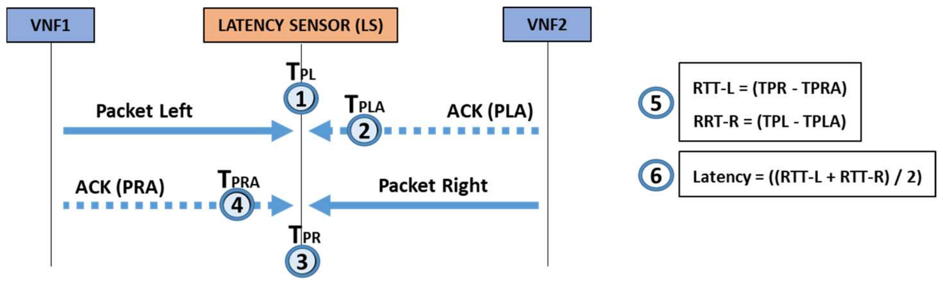


Fig. 2. Latency Sensing Mechanism.

In this way, by periodically taking samples of the inter-VNF data traffic and then analysing TCP packets, the sensor is able to calculate latency between VNFs at a specific service and to provide of this real-time latency information to upper layer elements (e.g., controllers, orchestrators, MANO, NFV-C). With this purpose, and during the whole lifecycle of the slice, it will continuously dump this data to a local repository so it becomes available to consumers via an external management network.

B. **Sensor Allocation**

As introduced in the previous subsection, the sensor is strategically placed between VNFs of a particular service. Then, it is important to mention first, that all the required code and configurations from the latency sensing mechanism are implemented in a VNF, so they can be deployed in this form at the corresponding Network Slice Instance (NSI).

In a multi-segment scenario, a NSI can be partitioned into a set of Network Slice Sub-Network Instances (NSSI) [11] following the slice composition concept given by the 5G-PPP in [4]. This concept, in particular, entails building a slice out of individual slices, then representing the allocation of smaller slices per network segment and the interconnection between them in a way to build a complete end-to-end slice.

The latency sensor in turn, should be allocated at a specific NSSI so it can fit in the VNF service chain. This allocation is normally performed at the slice provisioning stage so the sensor is deployed and interconnected along other service related VNFs. However, it could also be considered the addition of the sensor during the slice maintenance stage. In this case, one of the NSSIs corresponding to the service must be modified at runtime. The details on how the NFV-C modules help in coordinating the latency sensor provisioning are furtherly given in our previous work in [8].

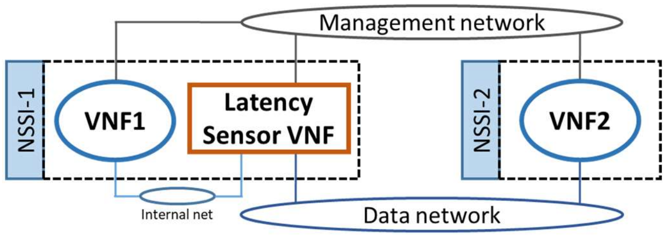


Fig. 3. Latency Sensor Allocation.

An example of latency sensor allocation is given in Fig. 3. In this case, the NSI corresponding to a service is composed by two VNFs and is partitioned into two smaller slices, NSSI-1 and NSSI-2. As depicted in the figure, the sensor is added to NSSI-1 so it can fit in between VNF1 and VNF2. It is important to consider that these VNFs are provisioned in different network segments, due to resource availability or to location constrain related to a particular service functionality. The sensor in turn, connects to VNF1 via an internal network and to VNF2 via the data network, so it can analyse data traffic flowing through. Moreover, both VNFs and sensor are also connected to the management network for external access, and in case of the sensor, to enable exposing latency data towards management components at upper layers.

IV. SENSING MECHANISM VALIDATION & ACTUATION CASES

To validate the latency sensing mechanism experimentally, we have set up a testbed following the principles introduced in the multi-segment architecture. In this way, the emulated scenario would allow deploying service slices along with the latency sensor to start gathering such data. Moreover, it would also enable testing preventive actuations in case service performance is not compliant with the subscribed Service Level Agreement (SLA). Fig. 4 depicts the set of management components and emulated network elements conforming the testbed, which in turn are distributed across five physical servers.

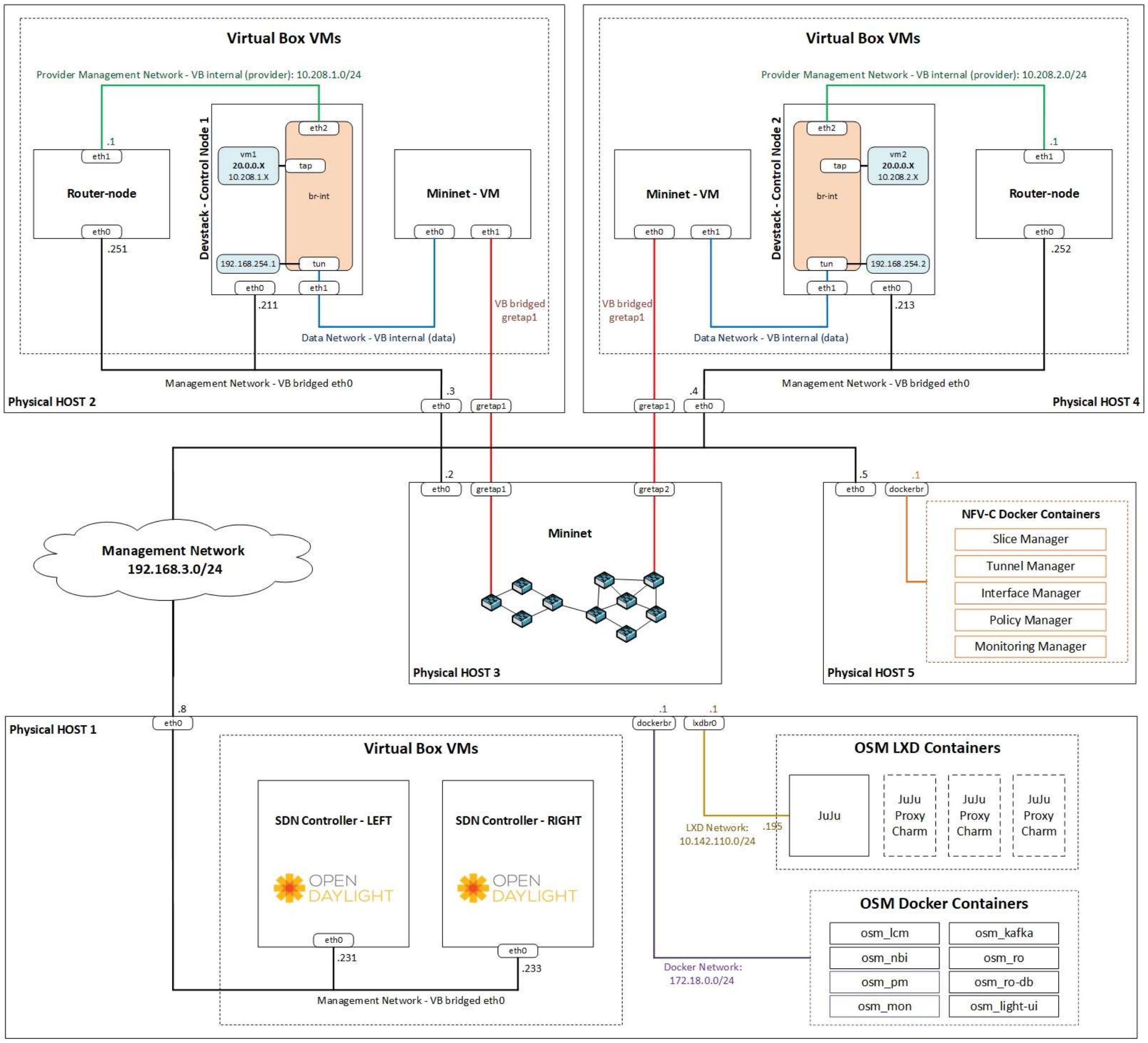


Fig. 4. Multi-segment experimental testbed for 5G service maintenance.

As shown, the testbed comprises a set of servers with the required software modules for slice maintenance. Particularly, the scenario represents two DC segments interconnected through the emulated WAN network, which in turn are controlled by high-level management entities such as VIM controllers, VIM orchestrators, the MANO component and finally the NFV-Coordinator. Server 1 allocates an instance of Open Source MANO (OSM) [12] and a pair of Opendaylight SDN controllers [13] for each of the emulated segments. Server 2 and 4 in turn, contain three components each, an instance of OpenStack orchestrator [14], a router-node VM for enabling external access to VNFs and a Mininet [15] instance to emulate local data centre network. The WAN then is emulated with Mininet at Server 3, and serves as the middle point connection between Servers 2 and 4. Finally Server 5 contains the NFV-Coordinator modules in the form of Docker containers. All servers are also connected via a Management network for inter-module messaging.

The next subsections detail the steps that were taken to validate the correct operation of the sensor and the gathering of latency data from running services, besides considering ways to guarantee their expected latency levels. At first, the sensor addition to the VNF service chain is analysed in order to validate its correct placement at the slice level and to test the calculation of real-time latency between VNFs. Then, the focus is set on describing how latency data is collected and stored at the NFV-C, and how it is used to provide awareness on the current state of configured datapaths. Finally, two actuation cases are examined to exemplify how the architecture is able to react by triggering policies, in case of latency thresholds violations.

A. <u>**Latency Sensor Validation**</u>

Considering the experimental scenario, the latency sensor is added to the end-to-end slice either at provisioning (i.e., by slice-composition) or during the maintenance stage (i.e., by slice-modification). As depicted in Fig. 5, the sensor now stands in a middle point between VNF1 and VNF2 and starts using the sensing mechanism to analyse the data traffic between them and then calculate the real-time latency. As the figure shows, VNF1 is connected to the LS via an internal Virtual Box network, while connection to VNF2 is achieved through the Mininet-based data network. Regarding inter-VNF data traffic, it is emulated using the iPerf network tool [16], setting a constant TCP data stream at 10 Mb/s.

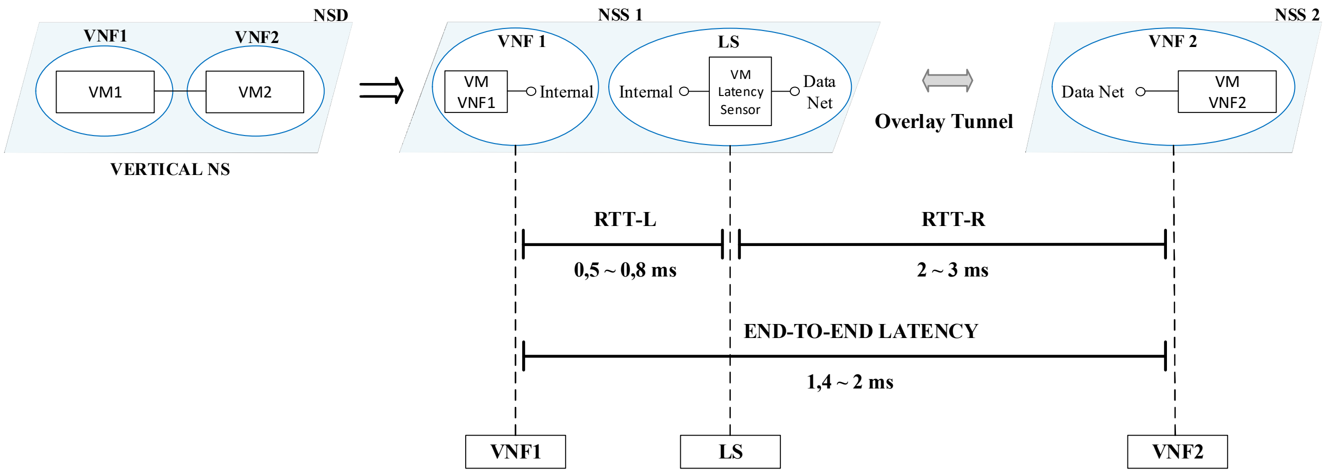
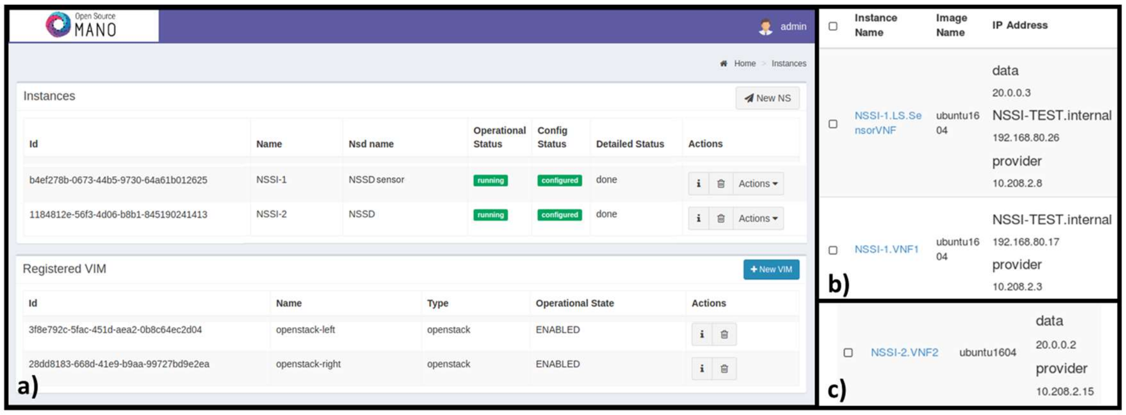


Fig. 5. End-to-end latency sensing in multi-segment allocated NS.

Once the sensor starts getting the time stamps data from the analysed messages, it calculates the RTT on each side. From the measures got at the experimental testbed, the RRT-L between VNF1 and LS shows times of around 0,5 to 0,8 ms considering that both VMs are deployed at the same OpenStack instance. On the other side, the connection between LS and VNF2 shows higher times considering a RTT-R between 2 to 3 ms, which are related to a higher delay in packet transmission at the overlay tunnel connection (through the data network) between the different OpenStack instances. With these numbers, the LS reaches overall results for end-to-end latency between VNF1 and VNF2 of around 1,4 to 2 ms. If these results are then compared to the time given by a direct ping latency test between VNFs without the presence of the sensor, it can be estimated that the time delay introduced by the sensor is about 0,2 to 0,4 ms, mainly due to the packet processing and redirection tasks. It is crucial to say that such values would vary when considering between testing over an emulated testbed and analysing latency in a real-case scenario, however and for the scope of this work, they serve to validate the correct latency sensor functionality.

Fig. 6 shows the scenario from a high-level point of view, more particularly, from the OSM (a) and OpenStack (b, c) dashboards view. In OSM the overall NS is depicted in the form of NSSI-1 and NSSI-2 instances, which correspond to the deployment of VNF1-LS and VNF2 respectively. Below it is also possible to see how openstack-left and openstack-right VIMs are registered so they become available for NS provisioning. In case of OpenStack, (b) shows the correct instantiation of the VMs comprising VNF1 and LS at the openstack-left VIM, while (c) shows the VM of VNF2 at the openstack-right VIM.



Instances

Id	Name	Nsd name	Operational Status	Config Status	Detailed Status	Actions
b4ef278b-0673-44b5-9730-64a61b012625	NSSI-1	NSSD sensor	running	configured	done	Actions
1184812e-56f3-4d06-b8b1-845190241413	NSSI-2	NSSD	running	configured	done	Actions

Registered VIM

Id	Name	Type	Operational State	Actions
3f8e792c-5fac-451d-aea2-0b8c64ec2d04	openstack-left	openstack	ENABLED	
28dd8183-668d-41e9-b9aa-99727bd9e2ea	openstack-right	openstack	ENABLED	

Instance Name	Image Name	IP Address
NSSI-1.LS.SensorVNF	ubuntu1604	data 20.0.0.3 NSSI-TEST.internal 192.168.80.26 provider 10.208.2.8
NSSI-1.VNF1	ubuntu1604	NSSI-TEST.internal 192.168.80.17 provider 10.208.2.3
NSSI-2.VNF2	ubuntu1604	data 20.0.0.2 provider 10.208.2.15

Fig. 6. Open Source MANO (a) and OpenStack (b, c) dashboards with deployed NSI.

B. **Link Latency Awareness**

Having validated the correct operation of the sensor in the previous subsection, the focus is then on how to retrieve and use latency information. In this regard, the MM module at the NFV-C is the one in charge of gathering such data from the sensors database via the management network, considering one deployed sensor per pair of VNFs whose inter-connectivity (i.e., datapath) is being monitored. Once this information reaches the MM level, it is sent towards the PM to check for policy compliance. Then, it is used to calculate the average latency for each particular datapath where a sensor exists, to be stored in a local database at the MM. The goal is to maintain an awareness of the current state of configured datapaths in terms of latency, so this information can be used in case any type of actuation is required to overcome detected degradations in the service performance levels.

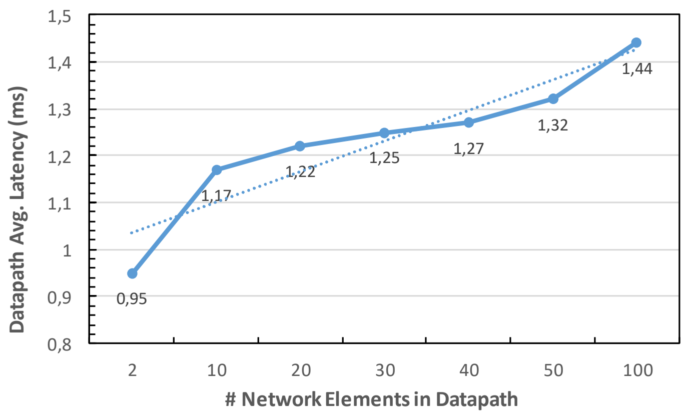


Fig. 7. Datapath Average Latency vs Number of Network Elements in Datapath.

Fig. 7 provides an example of the information stored at the MM database, where the values represent the average latency of monitored datapaths calculated using data retrieved from sensors, considering a set of datapaths with a different number of traversed network elements for each VNF inter-connection. In this way, if the PM determines that an actuation is required to guarantee compliant latency levels at a particular slice/service, it will consult this data to become aware of the state of configured datapaths and thus the current network state. As depicted in the figure, the configured datapaths average latency can be consulted by the PM, so it can furtherly consider this data to make a decision. The result could derive in actuation cases as the ones presented in next subsections, such as path reconfiguration and slice reallocation. These in turn, entail using monitored data to select the best way to guarantee service latency performance, considering required configurations to be triggered.

It is important to consider that this information should be continuously updated by the MM in order to maintain real-time information of the current state of the network. For this, a set of latency sensors should be deployed along all running services/slices either at provisioning time or during the maintenance stage.

C. **Actuation Case - Path Reconfiguration**

Given the awareness of real-time latency at running services provided by sensors, it is possible to use this data to identify if a particular link/segment is being affected due to possible network congestion or failure, after detecting a rise on the monitored datapath(s) latency levels. To this end, data regarding other monitored parameters (e.g., Bit Error Rate, Throughput, VNF CPU/RAM consumption) would also be used [17]. As introduced in previous sections, the architecture managing the scenario would act upon these types of events and trigger reconfigurations (i.e., actuations) over network elements to guarantee expected service functionality. In this case, Fig. 8 illustrates a conceptual scenario where service performance levels are maintained through path reconfiguration, upon the event of a recognized high latency on its currently configured datapath. The analysed actuation entails switching to an alternative datapath by reconfiguring network elements at the data layer.

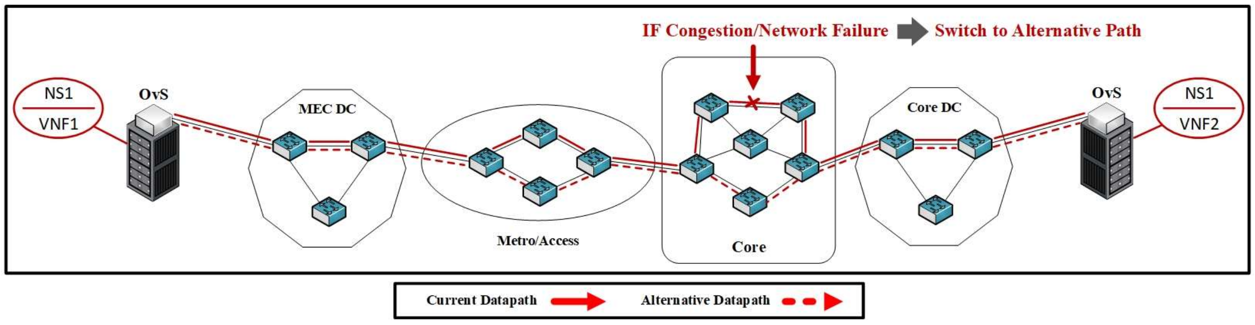


Fig. 8. Service Maintenance through Path Reconfiguration.

Fulfilling this actuation requires a set of given interactions and messaging between management components at different levels. To clarify this process, the actuation workflow for this specific case, depicted in Fig. 9, is described from a conceptual point of view:

- *Step 1 - (PM <-> MM):* During the continuous gathering of latency data, the PM is able to check whether these levels satisfy the ones defined at the service policies. In case thresholds defined are violated in a given datapath and a problem is identified at an affected link/segment of the network, PM sends a request to the MM to check the state of configured datapaths considering their average latency, and uses this data to compute an alternative path for this specific end-to-end connection.
- *Step 2 - (PM -> TM):* If an new path is selected as a possible solution for the event, PM triggers path reconfiguration by requesting the TM to configure the new datapath.
- *Step 3 - (TM -> WAN-Controller):* The TM is then the one in charge of contacting the WAN-Controller of the network segment in turn to give these instructions.
- *Step 4 - (WAN-Controller -> Network Elements):* The new route gets pushed to the network elements via southbound protocols, configuring the alternative datapath and deleting original configurations afterwards, in a way to avoid service disruption. Once all is set, the new path becomes operative and latency levels are expected to stabilize.
- *Step 5 - (MM -> PM):* In order to test the correct application of the actuation and as a step to reload again the sensing-actuation cycle, the MM restarts the continuous gathering of latency data from the sensor, besides the checking of its compliance with the set policies via the PM. In case any new value surpasses the permitted levels, the process starts over from Step 1.

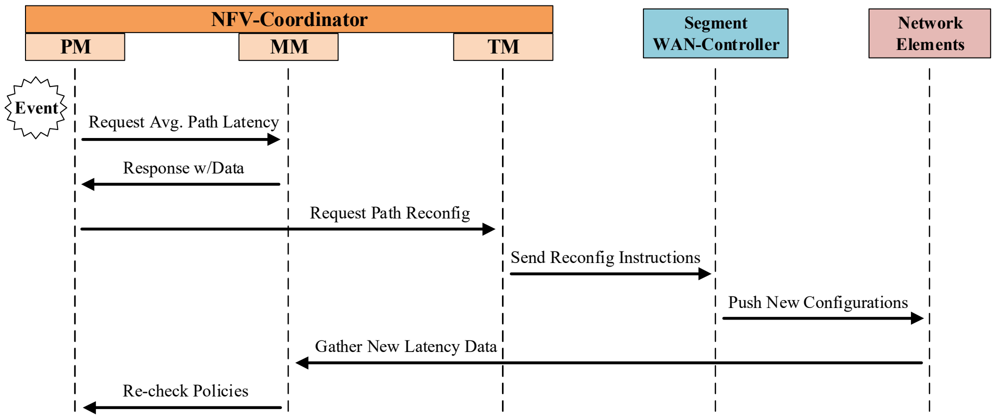


Fig. 9. Inter-module messaging workflow for path-reconfiguration actuation process.

D. **Actuation Case - Slice Reallocation**

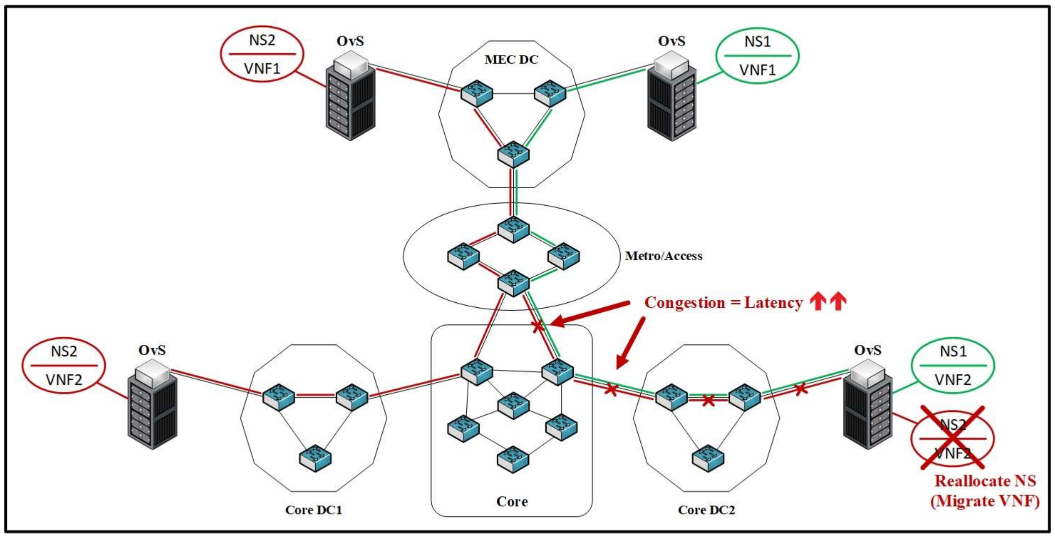


Fig. 10. Service Maintenance through Slice Reallocation.

A different case of actuation considers the reallocation of a particular slice/service to accommodate the network for better performance in terms of latency. Fig. 10 depicts a scenario that serves to exemplify this case from a conceptual point of view, focusing on the behaviour analysis of high-level management components. In turn, it shows two configured services with two VNFs each connected from Multi-access Edge Computing (MEC) DC to Core DC2 segment. The ones at the latter segment share the same physical location which in turn could potentiate the possibility of congestions when there is a large amount of traffic flowing to this area of the network. Once this event is recognized, an evaluation of other locations to check for available resources for VNF migration is conducted. As shown in the figure, the actuation entails migrating VNF2 of service 2 to Core DC1 segment in a way to distribute the traffic flow and guarantee expected latency level for both running services. It is important to consider here that performing a VNF live migration could introduce certain service downtime depending on how the process is executed. In this regard, the presented work focuses mainly on covering the triggering part of the process, are there already exist different analysis and mechanisms in the literature proposed to mitigate this effect [18-19].

This process requires more high-level operational steps at management components than the previous case. Fig. 11 helps to depict them, while a detailed description follows:

- *Step 1 - (PM <-> SM):* In case latency levels at a given datapath reach a state not compliant with the defined policies, PM would consider slice configuration and state of other monitored datapaths to analyse if the traffic flowing to VNFs is representing an overutilization of certain network sections (i.e., network congestion). Then, it will send a request to the SM to check the possibility of reallocating the affected slice (i.e., by VNF migration). A response would provide an option for a possible location (if it exists) to migrate VNF(s), considering computational resources availability.
- *Step 2 - (PM <-> MM):* The PM would then request the MM for the data regarding the current state of configured datapaths, in terms of average latency, so it can use it to compute the best candidate path towards the new considered location.

- *Step 3 - (PM -> SM):* In case an option is found for slice reallocation, PM sends a request to the SM to trigger the actuation.
- *Step 4 - (PM -> TM):* In parallel, PM requests the TM to configure the new datapath towards the new location, while considering as well deleting the previous one.
- *Step 5 - (PM -> IM):* IM should also be aware of the changes in order to set up routing configurations to and from newly placed VNFs at corresponding VIMs.
- *Step 6 - (SM -> OSM):* Once SM knows about the changes in the slices configuration, it sends specific instructions to OSM to fulfil them.
- *Step 7 - (OSM -> VIM-Orchestrators):* OSM then pushes these configurations to the involved VIM orchestrators, so these can proceed with the VNF migration.
- *Step 8 - (TM -> WAN-Controllers):* The TM contacts the WAN controllers of the correspondent network segments to give them the new configuration instructions.
- *Step 9 - (WAN-Controllers -> Network Elements):* Route(s) towards the new location(s) of migrated VNFs get pushed to the network elements via southbound protocols, configuring datapaths and deleting original configurations afterwards.
- *Step 10 - (IM -> VIM-Controllers):* The IM in turn, contacts the controllers of the correspondent VIMs to give them new instructions for inter-VNF routing. Once everything is set datapaths become operative and the latency levels are expected to stabilize across the involved segments.
- *Step 11 - (MM -> PM):* In a way to test the correct application of the actuation and as a step to reload again the sensing-actuation cycle, the MM restarts the continuous gathering of latency data from sensors and checking its compliance with set policies via the PM. In case there is any new value surpassing permitted levels, the process starts over from Step 1.

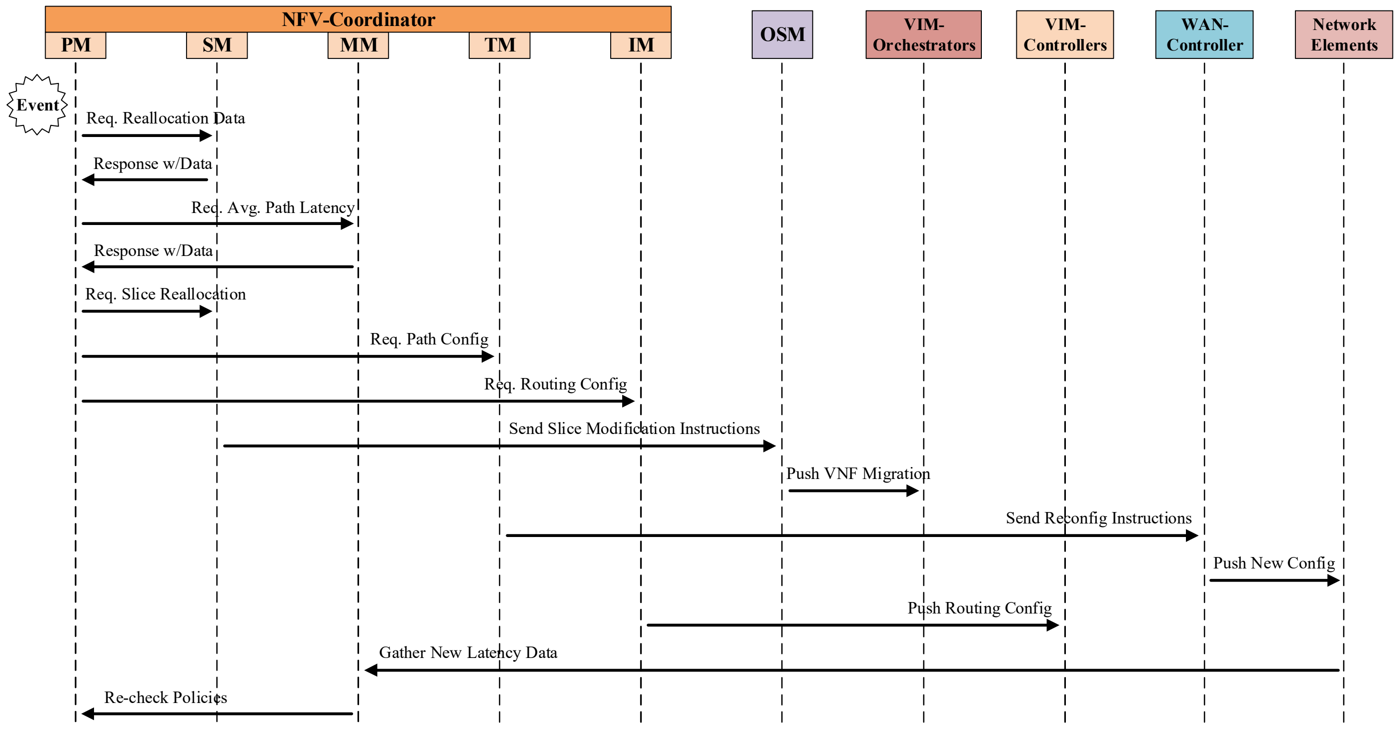


Fig. 11. Inter-module messaging workflow for slice-reallocation actuation process.

V. CONCLUSIONS

Considering the highly demanding requirements brought by 5G services, current networks need to be enhanced to fully support these stringent demands. With this purpose, new technologies such as Network Slicing have risen. The presented work bases on the use of this technology to provide a way to address one of the most relevant service type requirements, latency. In this sense, a multi-segment architecture is provided along with a mechanism for latency sensing which is deployed in the service chain in the form of a VNF. Results provided validate the correct latency data gathering and exemplify actuation use cases to prove the role of the architecture in guaranteeing latency-sensitive 5G services.

ACKNOWLEDGMENT

This work has been supported by the H2020 5GPPP SLICENET project (H2020-ICT-2016-2/761913) and the Spanish Government through project ALLIANCE-B (TEC2017-90034-C2-2-R) with FEDER contribution.

REFERENCES

[1] NGMN Alliance 5G White Paper, Version 1.0, February 2015.
[2] ETSI GS NFV-MAN 001 V1.1.1, December 2014.
[3] ONF TR-526, Applying SDN Architecture to Network Slicing, Issue 1, April 2016.
[4] 5G-PPP 5G Architecture White Paper, Version 2.0, December 2017.
[5] F. J. Moreno-Muro et al., "Latency-Aware Optimization of Service Chain Allocation with Joint VNF Instantiation and SDN Metro Network Control," 2018 ECOC, Italy, pp.1-3.
[6] S. Fichera et al., "Latency-aware resource orchestration in SDN-based packet over optical flexi-grid transport networks," IEEE/OSA JOCN, vol. 11, 4, pp. B83-B96, 2019.
[7] M. Szczerban et al., "Real-time Control for Deterministic and Dynamic Networks," 2019 European Conference on Optical Communication (ECOC), Dublin (Ireland), pp. 1-3.
[8] R. Montero et al., "End-to-end Network Slicing in Support of Latency-sensitive 5G Services", 2019 ONDM, Athens (Greece), 13-16 May 2019.
[9] T. Neagoe et al., "NTP versus PTP in Computer Networks Clock Synchronization," 2006 IEEE International Symposium on Industrial Electronics, Montreal, pp. 317-362.
[10] A. Pagès et al., "Dynamic Service Reallocation in NFV-based Transport WDM Optical Networks," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018.
[11] 3GPP TR 28.801, Study on management and orchestration of network slicing for next generation network, Version 15.1.0, January 2018.
[12] Open Source MANO, https://osm.etsi.org.
[13] OpenDaylight, https://www.opendaylight.org.
[14] OpenStack, https://www.openstack.org.
[15] Mininet, https://mininet.org.
[16] iPerf Measurement Tool, https://iperf.fr/.
[17] R. Montero et al., "Actuation Framework for 5G-Enabled Network Slices with QoE/QoS Guarantees," 2019 ICTON, Angers, France, 2019, pp. 1-4.
[18] N. T. Khải et al., "Optimising Virtual Network Functions Migrations: A Flexible Multi-Step Approach," 2019 IEEE NetSoft, Paris, France, 2019, pp. 188-192.
[19] K. Sugisono et al., "Migration for VNF Instances Forming Service Chain," 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, 2018, pp. 1-3.