ITUPublications

International Telecommunication Union

Standardization Sector

Technical Specification

(07/2020)

ML5G-DEL5

Vertical-assisted network slicing based on a cognitive framework

Technical Specification

Vertical-assisted network slicing based on a cognitive framework

Summary

This Technical Specification proposes a new framework that enables vertical quality of experience (QoE)-aware network slice management empowered by machine learning technologies.

Keywords

Cognition, network slicing, ML, NSp, NSsp, NSsu, QoE, QoS.

Note

This is an informative ITU-T publication. Mandatory provisions, such as those found in ITU-T Recommendations, are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

Contributors: Slawomir Stanczak
Fraunhofer HHI
Germany
Email: slawomir.stanczak@hhi.fraunhofer.de

.

Table of contents

Technical Specification

Vertical-assisted network slicing based on a cognitive framework

1 Introduction

Network slicing is a primary approach in future networks including 5G to meet the diverging services' requirements from diverse business verticals by creating multiple logical networks with the required quality of service (QoS). Furthermore, it is highly desirable that verticals are allowed to customise their network slice - based services and provide the customisation requirements as inputs in both the service provisioning and runtime phases. Upon receiving such vertical feedback, the management system of a future network e.g., 5G should apply the feedback to modify the service, and the corresponding end-to-end network slice for that vertical. Depending on the level of the verticals' feedback, the management system may need to convert quality of experience (QoE) requirements into QoS constraints and service level agreement (SLA) terms. Subsequently, these QoS/SLA constraints are employed to influence the modification of the network slice instance and/or inform other related operations to support fulfilment of the requirements in terms of QoS constraints. For autonomous network control and management, the above steps should be automated in a cognitive framework, enabled by machine learning (ML), automation and other related technologies. This document focuses on this approach and presents technical details especially related to the following aspects:

- Overall architectural design for this approach.
- Network slice service user (NSsu) (vertical is a part of NSsu) inputs and feedback into the service and corresponding network slicing management system.
- Mapping from the QoE requirements of the NSsu to the network slicing QoS configuration by the Network slice service providers (NSsp) and network slice providers (NSp).
- APIs for data monitoring and actuation to complete this cognitive loop.

In addition, a specific eHealth use case is described to provide a concrete example of this approach.

2 Overall architecture

A layered model for the different business roles that may be participating in the deployment, negotiation/advertisement and exploitation of multi-domain network slicing is introduced and elaborated. The followed approach considers the fact that technology domains are addressing network slicing from different perspectives based on the principles by which each domain is characterized and proposes abstraction practices for the provision of end-to-end services tailored to the heterogeneous needs of the network slice service user (NSsu). The differences among the technological domains with respect to slice representation are considered highly complementary for the support of a wider slicing concept which:

- Utilizes wireless technologies such as 4G and 5G.
- Supports dynamicity and adaptability with respect to the varied and distributed demands of the served Nssu.
- Allows for the expression of a multitude of performance, security and other requirements.
- Enables various stakeholders to be able to participate as providers.
- Allows verticals to have a certain level of control over their services for applying the principles of their application logic.

In order for this concept to be fully elaborated, it is required that the network slice conceptions of each technological domain should not be accounted for as the overall model or solution but only as a part of the overall concept applied to the proper partition or layer of the SliceNet slice where it is

applicable. This will allow the full exploitation of the offerings that a technological domain can expose. In most cases the slicing features that a technological domain can offer, are simply enablers for the provision of the slicing concepts of another domain. Finally, but most importantly, it is required to understand that vertical needs are not expressed in terms of a technology related nomenclature but, on the contrary, in terms of application specific semantics that neglect or at least tend to put aside what each technological domain can offer.

A set of network, resource and function abstractions to address the need for harmonisation of the management processes over the underlying resources that can be considered a first level of offering exposure is defined. These offerings regard parts of radio access and/or core network functionalities that can be supported by a network slice provider (NSp) in the context of a network slice provisioning request. Apart from the functional role of these offerings, the need for quality based management and SLA enforcement requires that additional information relating to the operation (performance, faults) of the allocated resources or the delivered data services (achieved data rates, bit error ratio, latency, etc.) as well as potential actuation options should be also made available for utilisation in higher level workflows. The level of provisioned support of these monitoring and actuations options by the underlying resources is subject to a selection refinement process of the offered resources from the NSp which has to combine (and configure) resources, either physical and/or virtual, to build 4G/5G communication services and additionally activate its own business added value services that aim at the continuous evaluation and adjustment for the maintenance of the intended levels. This will result in a network slice, with a specific set of requirements, that the NSp will advertise to the NSsp (network slice service provider).

The NSsp in turn will compose an end-to-end network slice, which will be the composition of one or more network slices provided by several NSp, and will advertise these end-to-end network slices in the form of digital services to the network slice service user (NSsu) or more commonly referred to as the vertical. The digital services and their requirements define the context of the vertical role with a specific business interest. The three roles (NSp, NSsp and NSsu) interact on the basis of a producer-consumer relationship.

3 Vertical QoE feedback

During the runtime of NSs, it is important to guarantee that the quality of the applications and services executed on top of them is at optimal levels, without any degradation. In this process, it becomes essential to gather the experienced quality from the NSsu perspective, as they are the ultimate users of the deployed NSs to which both control and management systems should cater for their needs and requirements. In order to retrieve the NSsu perspective about the current quality and incorporate it into the decision-making of the management and control systems, a feedback mechanism that allows the expression the experience of vertical customers (i.e., NSsu) about the provisioned service is proposed. This mechanism aims to progress towards the inclusion of the verticals customers' in the whole management/control of the deployed slices (Figure 1). Thanks to the capabilities exposed through the OSA access layer, which grants a vertical customer access to the underlying management and control ecosystem, this feedback mechanism allows collecting a qualitative or quantitative rating – a mean opinion score (MOS) value – from the NSsu through a dedicated function enclosed within a specialized plugin (QoE plugin) in the P&P controller of the slice (Step 1). The collected rating can then be captured by the corresponding instance of the QoE optimizer (Step 2b) which has several policies in place that dictate under which event and condition what action (ECA) should be carried out to remedy bad quality situations. In the case of NSsu feedback, the policy would state that, given a specific MOS value (e.g., "Bad"), the QoE Optimizer should trigger the specified (re-)configuration operation towards the corresponding enforcement point for the operation (i.e., the network slice service provider orchestration system) so as to maintain optimal quality levels (Step 7). This policy is initially distributed from the Policy Framework (PF) upon instantiation of the slice and all software artefacts responsible for its control and management. The parameters of the policy (i.e., the ECA

items) would be configured according to the requirements of the slice and the service it needs to support.

Such a policy system allows the decoupling of the "what" from the "how" of the defined actions: the QoE optimizer is the entity responsible for knowing the type of action that needs to be enforced under certain stimulus conditions (the event), but it does not have the knowledge neither the capacity or the tools to execute the several operations required to materialize the action (i.e., "the how"). This capacity is assumed by the enforcement point, such as the orchestration system, which has the full view of the underlying network infrastructure, its technological details and available resources as well as the capacity to coordinate all the operations through the layered infrastructure to achieve the desired (re-)configurations stated by the action parameter.

While the expressed feedback value states the degree of quality from the NSsu perspective, it does not necessarily indicate the underlying parameters of the configured NSs which have a greater influence on the experienced quality at the user side. In fact, the feedback provides an opinion-based view about the service provided, while the monitoring of the underlying slices and network is focused on measurable QoS KPIs, such as bit error rate, latency, jitter, etc. As such, it becomes essential to tie the expressed feedback (i.e., the QoE) with the parameters that can be monitored at the slice/network side (i.e., the QoS). Clause 4 elaborates on the aspect of QoE/QoS mapping. Then, thanks to this mapping, it becomes plausible to identify the slice/network parameters that can be re-configured to influence the quality as it is perceived by the NSsu. Indeed, by setting up policies which tie the vertical's feedback with an action which (re-)configures the most relevant QoS parameters that are affecting the expressed QoE, it is possible to fully realize the "verticals-on-the-loop" vision that 5G is advocating. The inclusion of vertical in the control loops requires that feedback options are expressed in the context of the vertical business. This in turn is based on the fact that NSsp resources are offered in a technology abstraction manner that hides underlying complexity and exposes only a number of properties (sensing or actuation) from a predefined set (reference set of definitions) onto which the underlying capabilities can be mapped. In this way the vertical can see a set of available options that are relevant to its business without knowledge of the mechanisms that realize them. Such options allow also different vendors to provide their enabling artifacts on the condition that these adhere to the predefined set of defined sensing, actuation and optimisation options so that, once selected, dependencies can be resolved to allow the artefacts to operate as expected. To fully realize the concept, the proper analysis of the received QoE feedback is required in order to influence the policy system and the related actions to apply the most suitable (re-)configuration according to the current slice status. Starting from the QoE plugin, which has certain dependencies on input types, the collected MOS value would need to be inserted into the shared time-series-based data lake (Step 2a). Upon insertion in the time series database, a specialized analytical function/ML model would collect the value along with the QoS metrics of the affected E2E slice that are maintained in the data lake according to the vertical's requirements during slice ordering (Step 3) in a periodic manner to perform background analysis of both types of information. The ML models are located within an ML pipeline, in which data aggregation/preparation modules pre-process the data so that it is ready for consumption by the models as explained in Step 3. Thanks to the capabilities of the P&P control, the ML pipeline and the ML models can be plugged into the slice specific control, along with other plugins, such as the QoE plugin, to perform specific analysis related to the type of slice that is deployed and the services supported over it. For the particular case of the feedback mechanism, this monitoring dependency is resolved from the high-level requirements, selected as options by the vertical, in the opposite order to that which used to make the options available. In this resolution process and based on the involved domains the abstracted technology artifacts are engaged for the particular slice and their monitoring output is made available as metric types that the higher-level components can process. The goal of such a model is to analyse the expressed feedback and the monitored QoS at the slice level, in order to identify the main cause(s) for unsatisfactory quality levels (Step 4). After this analysis, the model would then recommend to the PF which QoS parameter(s) need to be influenced as part of policy actions (Step 5). Given these gained insights, the PF would then update the action

field of the policies related to the expressed vertical feedback for slice quality maintenance, i.e., the QoE optimizer (Step 6). With this updated policy, the QoE optimizer would then be able to trigger the most suitable action according to the most updated insights gained through the expressed feedback from the vertical customer (NSus) once received, following the same procedure as in Step 7. Figure 1 illustrates the NSsu feedback mechanism.

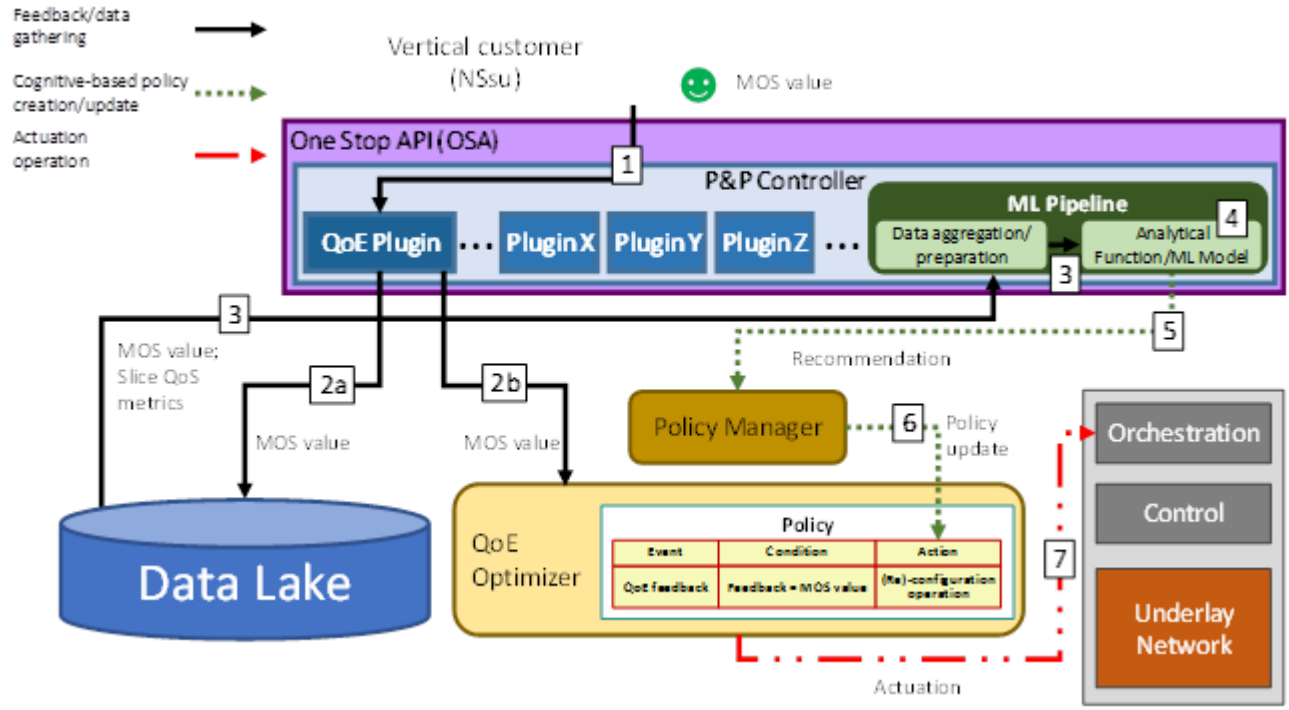

Figure 1 – NSsu feedback mechanism

Table 1 summarizes the steps involved in the feedback mechanism, providing additional relevant details for each step:

Table 1 – Vertical QoE feedback mechanisma

Step	Description	Additional details
1	Thanks to the capabilities exposed through the QoE plugin, the vertical customer expresses its feedback about the quality of experience.	The feedback takes theform of a MOS value, for instance: • Scalar: 1, 2, 3, 4 or 5 • String: very bad, bad, regular, good, very good
2a	The feedback is inserted into the shared data lake for its later consumption by ML pipelines.	A specific schema in the data lake should be employed for this purpose, specifying for each entry, at least, the time of collection, the value of the feedback and the identifier of the slice for which the feedback has been collected.
2b	The feedback is directly consumed by the QoE optimizer for immediate actuation to maintain quality levels.	As a pre-condition, a policy should already have been instantiated at the QoE optimizer specifying the ECA values for the actuation related to the feedback.

Table 1 – Vertical QoE feedback mechanisma

Step	Description	Additional details
3	The feedback from the NSsu and the slice QoS parameters are gathered from the data lake by an ML pipeline to perform their analysis and derive insights.	The QoS parameters are aggregated and prepared per end-to-end slice by aggregation modules of the monitoring system. Then, pre-processing modules within the ML pipeline further prepare the data to be ML ready, such as through the normalization and smoothing of the data. As such, the ML model would consume the data from the data aggregation/preparation modules within the pipeline, which already gathered from the schemas of the data lake the information that is tagged under the identifier of the slice to which the ML model belongs.
4	The ML model, as a pluggable function of the slice control, performs the analysis of the consumed data.	The analysis done by the ML model is the responsibility of the ML provider, be it the same NS provider or the vertical customer. The P&P framework only provides a platform that allows the dynamic instantiation of pluggable control functions per slice.
5	After the analysis, the ML model contacts the policy framework to recommend the QoS parameters of the underlying slice that need to be (re-)configured in case undesirable QoE situations are reported through the feedback mechanism.	The Recommendation should indicate which of the analysed QoS parameters of the slice gathered from the data lake need to be stated as to be reconfigured in the action field of the policies related to the feedback event. To enable the exchange of information between the ML models and the policy framework, a common information model for the action should be in place, for example, as in [b-ITU-T Y.3174] or [b-ONAP].
6	The policy framework disseminates the updated policy about the treatment of the feedback from the NSsu to the QoE Optimizer in order to have the most up-to-date action related to actuation.	The policy framework and the QoE optimizer should have a synchronization mechanism for policy update, either by subscription or active polling from the QoE optimizer or active dissemination from the policy framework. Such a mechanism would enable the gathering of the latest policies related to the specific slice identifier to which the policy refers.
7	At the time of an actuation trigger due to the reception of a feedback value, if a policy is in place that ties the received feedback value to a concrete (re-)configuration to be carried out, the QoE Optimizer contacts the Orchestration system of the underlay network to apply the action.	The QoE Optimizer is only responsible for determining when and what of the actuation, while the how is the responsibility of the Orchestration system, which should engage with the Control of the underlay network to achieve the specified (re-)configurations with the specified parameters.

4 QoE/QoS mapping

4.1 QoE/QoS mapping

The cognitive sub-plane incorporates multiple ML based strategies to estimate network slice (NS) QoE from QoS in order to enable QoE-aware NS management. The estimations of QoE levels enables

slice providers to support the desired E2E QoE-based service level agreements (SLAs), e.g., by optimizing resource allocation through SliceNet actuators; furthermore, it may help troubleshoot issues in the provider's infrastructure.

The general assumption is that the NSsp and NSp can measure various QoS metrics; however, they do not have full information on the actual QoE that the user is experiencing. Therefore, they must estimate the QoE from the measured QoS metrics.

4.2 QoE -aware network slice management

The SliceNet cognition sub-plane provides a framework for the QoE-aware management of NSes on top of a shared 5G network infrastructure. QoE-awareness allows SliceNet to identify degradation of slice users' QoE by directly monitoring user feedback (see clause 3) or by inferring QoE from QoS derived from measured slice infrastructure metrics. Examples of user feedback are quality metrics transmitted from the UE, feedback about user satisfaction from the NSsu, etc. Examples of QoS are KPIs derived from infrastructure resources (CPU, memory, BER, etc.) and network traffic (latency, throughput, etc.). Two main developments have been key to achieving this objective: the development of ML analytical models to gain insights about the QoE of deployed slices and the development of an actuation framework devoted to the maintenance of quality levels per slice.

In SliceNet, multiple ML models serve as advanced QoE sensors that determine relevant metrics and KPIs to determine the quality of the underlying slices or events/changes on the infrastructure that may affect the health of the slice. In this regard, the SliceNet cognition sub-plane has contributed to the progress of QoE awareness/analysis of NSsu slices by analysing the requirements of selected NSsu UCs use cases and mapping their requirements onto metrics at the slice level that may affect their quality. This work can help with other UCs use cases related to similar NSsu sectors as samples of the metrics to be employed for QoE monitoring of slices supporting specific NSsu services. In addition, aside from the specific models developed for SliceNet's NSsu UCs use cases, SliceNet's cognition sub-plane also focused on the generic aspect of QoS to QoE classification, which is very relevant to optimizing QoE-aware management of NSes. Indeed, current Internet service trends (e.g., Youtube, Skype) usually employ a QoE rating to classify the quality of the service. Then, it is up to the service/network operator to determine, in case of bad quality, which network parameters are responsible for the degraded service (i.e., QoS metrics). Such a task is usually complex, as there is not a direct correlation between user QoE and network QoS metrics. Hence, the model developed within SliceNet's cognition sub-plane for this objective (the QoS to QoE classification model) sets a solid framework and methodology that could be expanded and contextualized to tackle the task of correlating QoE and QoS in other NSsu use cases.

4.3 Strategies to derive QoE from QoS

4.3.1 Estimating QoE from measured NS infrastructure QoS

In this strategy, the relationship between the target application's QoE and the QoS is learned during the training phase, but at run-time only metrics from the provider's infrastructure are collected to derive the QoS. This run-time QoS is used to infer the QoE from the training models to trigger remedial actuations when required. Our premise is that although the provider can only observe partial network information, the E2E QoE is exposed in these observations. With this goal in mind, we describe the scenario for the training of the ML model. Two slices that share infrastructure resources are required. One slice supports the target application, while on the other slice background network traffic is runto generate network congestion; since the slices share infrastructure resources the background traffic will interfere with the performance of the target application which will in turn affect the target application's QoE. Training data is created from simulations of various levels of traffic congestion running in parallel to the target application. While running these simulations QoS metrics are collected from the provider's infrastructure monitoring framework. At the same time QoE metrics are collected from the user application or through MOS based feedback from actual end users.

An ML model correlating the QoS features with the target QoE metrics is generated to estimate the QoE from QoS.

At run time, this QoE estimation model combined with current QoS measurements serves as a trigger for remedial actions by the SliceNet cognitive sub-plane. In cases of unfavourable QoE, policies defined within the sub-plane would specify which remedial actions need to be triggered when facing unfavourable QoE. These remedial actions would be communicated to relevant SliceNet functional modules (e.g., orchestration modules) to execute the desired (re-) configurations.

Figure 2 and Table 2 illustrate the workflow of the *estimating QoE from measured NS infrastructure QoS* strategy during run-time.

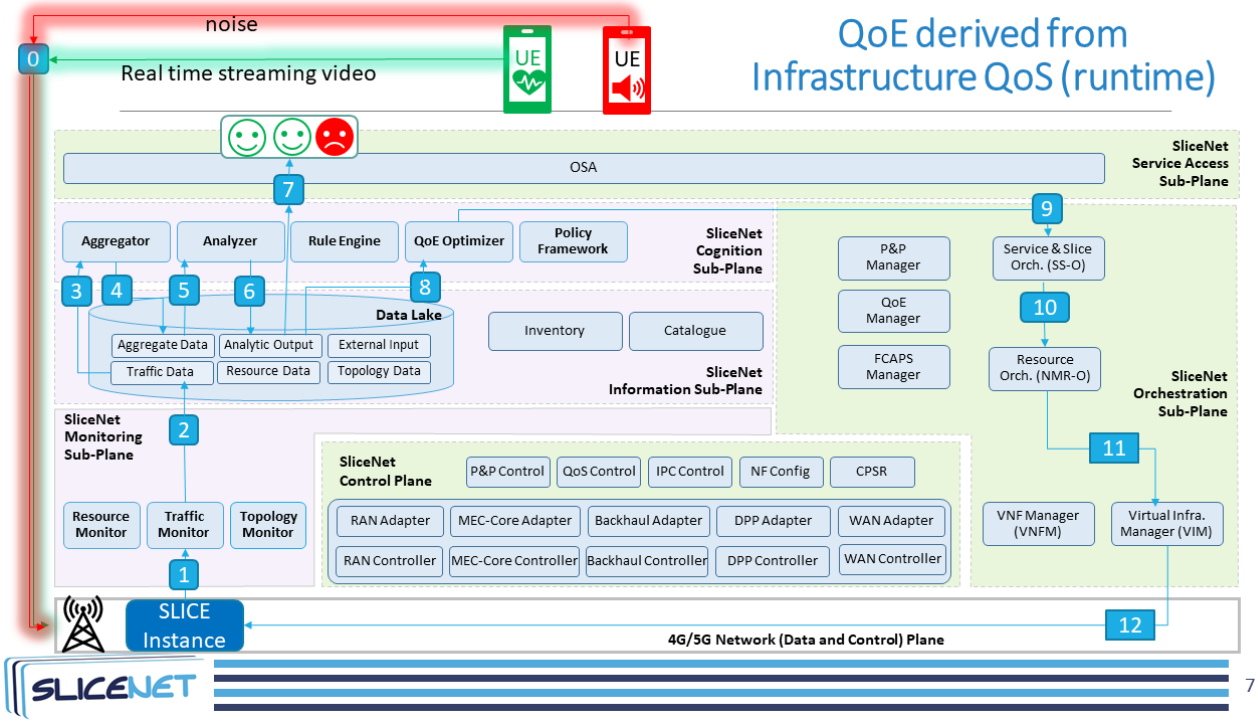

Figure 2 – Estimating QoE from measured NS infrastructure QoS (run-time)

Table 2 – Estimating QoE from measured NS infrastructure QoS (run-time)

Step	Description
0	A UE mobile device streams data into the network slice assigned to the target application. In the background, noise is being generated by other UE devices running background network traffic on another slice to generate network congestion.
1	The traffic monitor (and resource monitor) continuously collects network flow metrics related to a VNF interface servicing the stream.
2	The traffic monitor stores the collected metrics in the data lake.
3	This traffic data is elaborated by the aggregator, which prepares and transforms the data to make it ML-ready.
4	The aggregator inserts the aggregated QoS data into the shared data lake.
5	The aggregated data related to the performance of the deployed VNFs is collected by the Analyser from the data lake.
6	The analyser (i.e., the ML model) analyses the QoS data, derives the QoE classification of the target application, and produces a QoE classification event for the data lake.
7	Display QoE classifications over time.

Table 2 – Estimating QoE from measured NS infrastructure QoS (run-time)

Step	Description
8	The QoE optimizer, acting as the sole PDP, polls the data lake and collects the stored events related to the previous analysis.
9	When the QoE optimizer sees a series of unfavourable classification it then decides, based on policies delivered by the policy manager, the most suitable remedial action to overcome the undesired states of the affected VNF instance. It may decide on scaling overloaded VNFs or migrations related to quieting the noisy VNFs (lowering their priorities, reducing their allotted resources, etc.). The decision is delivered to the slice and service orchestrator.
10	The slice and service orchestrator delivers the decision to the resource orchestrator.
11	The resource orchestrator, given the forwarded request and its parameters, determines the extra resources that are required (upscaling case) or the new location (migration case) for the VNF instance for which the actuation has been triggered. To enforce the desired action, it engages with the virtual infrastructure manager.
12	The virtual infrastructure manager, being responsible for the management of the virtual computation resources (e.g., VMs), executes the determined action.

Figure 3 and Table 3 illustrate the workflow of the *estimating QoE from measured NS infrastructure QoS* strategy workflow during the ML model training phase. The main difference between the run-time and training phase is that the QoE input from the NSsu, i.e., UE quality metrics, is consumed in the training phase, but not at run-time; rather at run-time the QoE is derived from QoS.

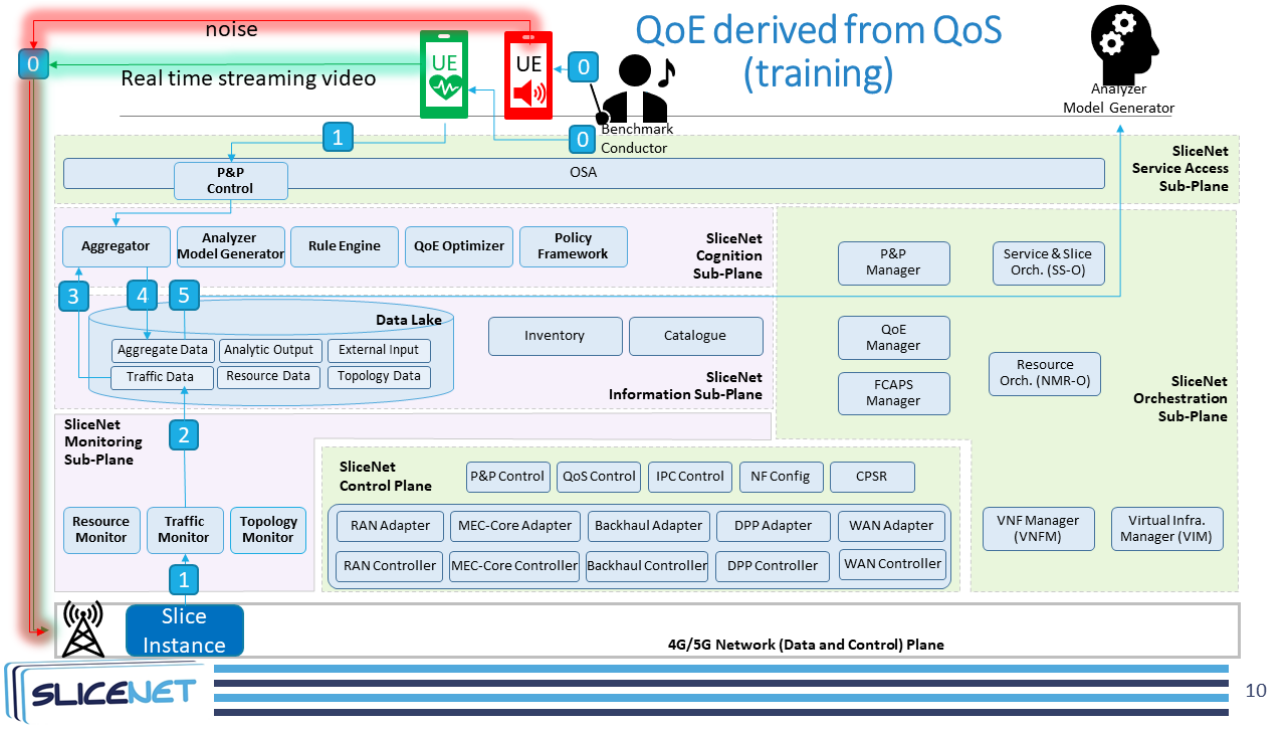

Figure 3 – Estimating QoE from measured NS infrastructure QoS (training)

Table 3 – Estimating QoE from measured NS infrastructure QoS (training)

Step	Description
0	The benchmark conductor runs many benchmarks. Each benchmark consists of a real time video stream with various levels of background noise (including no noise). Each benchmark is assigned

Table 3 – Estimating QoE from measured NS infrastructure QoS (training)

Step	Description
	a benchmark ID. The benchmark ID is used to map metrics from the UE's streaming video device and the network flow metrics from the slice's VNF.
1	The traffic monitor (and resource monitor) continuously collects network flow metrics related to a VNF interface servicing the stream. In parallel, the UE streams its quality metrics to the aggregator (thanks to the P&P control).
2	The traffic monitor stores the collected metrics into the data lake.
3	This traffic data and the UE quality metrics are elaborated by the aggregator, which prepares and transforms the data to make it ML-ready.
4	The aggregator inserts the aggregated QoS data and UE quality metrics into the shared data lake.
5	Once all the benchmarks are run, the analyser model generator: • Reads accumulated QoS metrics & UE quality metrics from data lake. • Derives target QoE Classifications from no noise UE quality metrics aggregations. • Derives QoS features aggregations highly correlated with QoE Classifications. • Generates ML model with target QoE Classifications from QoS features. • Persists the ML model. The ML model is used at run-time to derive QoE Classifications from the QoS metrics.

4.3.2 Estimating QoE from measured UE quality metrics

In this strategy, the target application's QoE is inferred from UE quality metrics. The QoE is inferred from UE quality metrics learned during the training phase to generate the QoE estimation model. At run-time the same UE quality metrics are collected and streamed into this QoE estimation model to trigger remedial actuations when required. Specifically, the SliceNet OSA P&P control enables the UE to transmit its quality metrics to the cognitive sub-plane's analyser which feeds the metrics into the QoE estimation model. The QoE estimation model then triggers the QoE optimizer and policy framework to decide on the proper remedial actuations.

For instance, the cognition capabilities that allow for the realization of anomaly detection for a real-time video streaming application could reside exclusively at the NSsp level. The NSsp is responsible for maintaining the quality of the streaming video connection to the provided slice in order to guarantee a healthy E2E service across the provisioned infrastructure. After collecting RAN-related information from the UE endpoint, the NSsp determines if an anomaly is going to happen at the connection channel that may affect the ongoing service. When an anomaly is detected, the NSsp triggers the necessary remedial actions to overcome the predicted anomaly.

Figure 4 and Table 4 illustrate the workflow of the *estimating QoE from measured UE quality metrics* strategy during run-time.

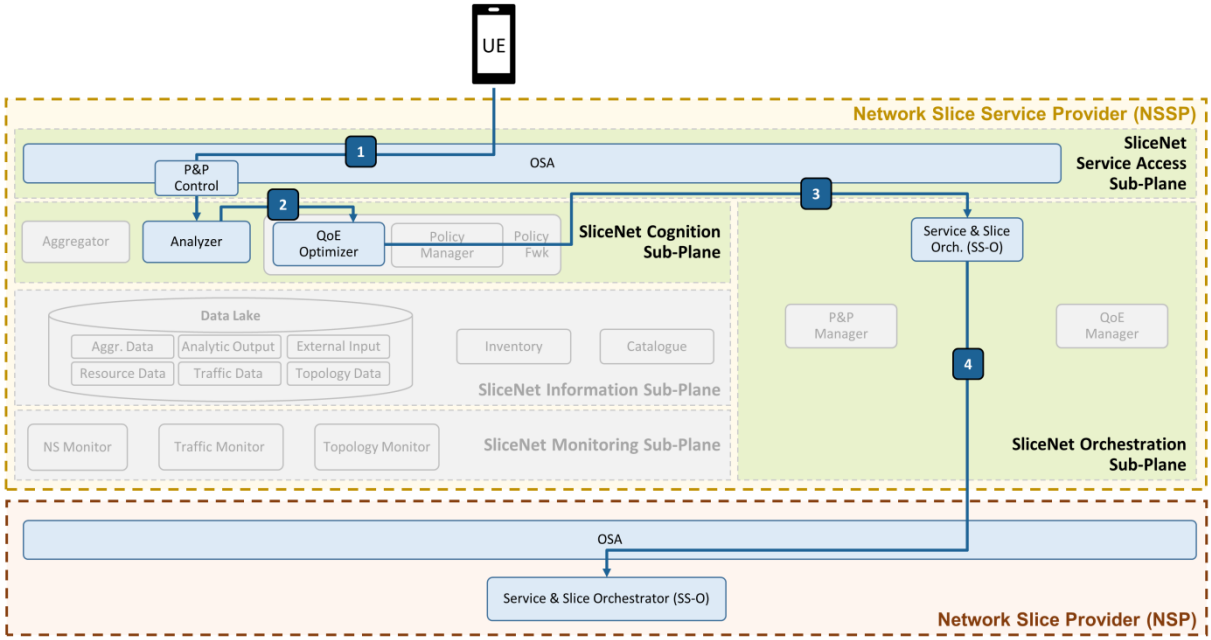

Figure 4 – Estimating QoE from measured UE quality metrics (run-time)

Table 4 – Estimating QoE from measured UE quality metrics (run-time)

Step	Description
1	Thanks to the capabilities exposed through the OSA and the P&P control, the information related to quality metrics from the UE is directly streamed towards the analyser.
2	The analyser (i.e., the anomaly detection ML model) analyses the data and produces an anomaly prediction event (if any) which is sent directly to the QoE optimizer.
3	The QoE optimizer consumes the event and decides, based on policies delivered by the policy manager, that in order to avoid predicted anomalies on the video streaming service, a handover is required. A handover means that the UE is re-associated with a healthier eNB or RAN slice within the NSp infrastructure and disassociated from the failing slice's resources. The decision is delivered to the slice and service orchestrator of the network slice service provider.
4	The slice and service orchestrator at the NSsp level engages with the slice and service orchestrator at the NSp level in order to enforce the handover/UE re-association. The exact action will be carried out thanks to NSp level capabilities (e.g., control plane functions).

5 Control plane and one-stop API

5.1 One-stop API

The SBA approach implied by the ongoing 5G new core specification in combination with the DECOR/MOCN/CUPS approaches for 4G, is paving the way, or at least providing room for assuming that in the near future operation practices will be based on a fully modular and softwarized approach. This assumption can be further elaborated when the question comes to the actual nature of the operator or better the NSp since the operation tasks can be segregated in self-contained parts which in turn can be utilized in more complex service assemblies spanning the borders of traditional administrative domains.

A network slice provider (NSp), therefore, can be an entity that invests in hardware resources and is able to provide services on top of any (or a combination) of the following offerings:

- (Radio) Access resources that can support slicing in terms of sharing of the access medium.
- Network (IP) connectivity that can support slicing in terms of isolation of network overlays.

- Data center resources that can support slicing in terms of isolation of running services that are offered as instances of onboarded images (VMs/containers).

Each of the above can be implemented by different technologies based on the investment decisions of the provider.

The vertical relationships imply several transactions among them that are performed in the context of slicing support. The above roles can be provided by different administrative domains separately or in combination. The transactions, therefore, are performed in the context of a contract among the administrative domains. Such a contract identifies the access rights of the consuming side over the interface exposed by the producing side. The consuming side, however, is always a set of artifact objects acting on behalf of a topmost end user (vertical) but in the context of the arrangements that have been put in effect in the process of service provisioning by the involved roles (NSsp and NSp).

The transactions identified above serve a single purpose: the provision of end-to-end communication services leveraging all the aspects of emerging technologies. Considering this fact, SliceNet proposes a common approach for the support of all the interactions via different projections of the information accessible at each point based on the role of the entity that consumes or influences the information. This translates into the definition of the concept of the one-stop API (OSA) as an approach that is expected to allow the creation of an ecosystem with easy participation of various players in the context of the business purposes of the related role.

In the above process, starting from bottom to top the following OSA transactions/projections are foreseen per role:

- NSp
  - actuation descriptor onboarding
  - monitoring descriptor onboarding
  - resource types registration associated with actuation and monitoring descriptors
  - identification of resource types availabilities
  - onboarding of policy-based rules with dependencies on actuation and monitoring descriptors
  - coverage (location) support
  - resources are listed based on type and capabilities
- NSsp
  - observes technology agnostic information of descriptors
  - identifies monitoring and actuation capabilities offered by the NSp
  - is able to select SLA related automation rules
  - is able to indicate which of the monitoring and actuation options should be activated per slice type
  - enhances management of underlying NSp supported monitoring and actuation options by crafting and registration of QoE, Cognition, and P&P service aspects
  - registration of slice templates with dependencies on QoE, Cognition, and P&P service aspects
  - slice templates are listed based on service tailored aspects and underlying dependencies including location support per aspect
  - slice templates are annotated with service features
- Vertical
  - observes slice templates through service features
  - selects service features to be activated

- gets access to slice specific actuations and slice monitoring as exposed via P&P functions

Vertical requests trigger the NSsp to provision an end-to-end network slice through a multi-level orchestration process. Along the instantiation of the artifacts at each level to support the deployment and operation of the slice, the descriptors are instantiated on the basis of the resolution of the provisioning details to form an inventory distributed across the involved domains. Inventory entries created shall be sufficient to allow management modules per domain to align with the required support for monitoring, actuation, cognition and QoE processing as well as the support of the vertical tailored view of the slice via the P&P approach.

5.2 P&P framework

The P&P framework aims at providing truly customized runtime control, management and operation of end-to-end slice instances while offering vertical-tailored services. It enables slice customization, as it provides a novel combination of tailored control functions, APIs and tools that are offered to verticals to let them specialize their slice instances according to their needs.

In practice, the P&P framework provides an innovative control environment, dedicated per slice, which offers the verticals direct control of slice instances and provides a new flavour of customization. The main idea is that the P&P framework provides, for each slice instance, a vertical-tailored abstraction of the slice (including its view and control exposure) on top of the control plane APIs. It is composed of the combination of:

- A slice view abstraction, that offers an abstract and vertical friendly view, on top of which the logic for the control operations is added so that it can be consumed by verticals at runtime over their slices. This is done by means of a dynamic and automatic mapping of operations issued from the vertical oriented APIs into specific actions to be enforced or performed through the pluggable control functions.
- Vertical-oriented APIs, that provide the set of tailored control and management APIs that are exposed to the vertical and offer the runtime control operations over the slice-specific model and view. The verticals access these APIs through the one-stop API framework.

These P&P vertical-oriented APIs rely on a generalized slice exposure model that enables a vertical customized abstraction of slice instances, still agnostic to the slicing technology and able to address the description of any level of managed entity from vertical service, to slice, and down to network functions and resources. This exposure model provides a slice topology view that is augmented with a set of runtime control operations that are offered to customize the slice control, management and monitoring: these are in practice the customized control primitives at the disposal of the vertical.

The P&P vertical-oriented APIs are defined as the whole set of REST APIs that a given P&P control instance exposes for enabling tailored runtime control to verticals. These REST APIs can be grouped into two main categories:

- **Continuous monitoring APIs**, to expose towards verticals (through the one-stop API) continuous data streams including tailored slice performance metrics and statistics.
- **Runtime control APIs**, which are the collection of tailored control APIs exposed towards verticals and dynamically created according to the customized slice view. These APIs provide a further abstraction on top of the control plane APIs.

5.2.1 Continuous monitoring APIs

The **continuous monitoring APIs** provide access to slice monitoring information and can be considered a basic control exposure feature allowing verticals to be aware of statistics and performances related to their slices. The nature of the monitoring data accessible is subject to the P&P control customization process, and all kind of resource, network function, slice or service level monitoring information can be in principle exposed to verticals according to the agreed control exposure. The P&P framework exposes slice related monitoring information as continuous streams

of data. The base idea is to transform single-query monitoring operations offered, usually an HTTP GET, into a periodic-executed query that pushes results on a proper channel, creating a data stream. The technology used to create such a channel is based on the concept of WebSocket [b-WebSocket]. Each P&P instance is able to handle one or more websocket connections in order to meet the requirement of having multiple streams per slice. Once the websocket is opened, a vertical can request through the one-stop API the continuous monitoring of a desired performance metric (e.g., through a graphical user interface), on condition that such a metric can be retrieved by a GET runtime control operation already exposed by the given P&P instance. In practice, a vertical can request the P&P instance to periodically execute a given monitoring query and then collect the results through the opened websocket as continuous data streams.

The vertical, through the one-stop API, can perform such continuous monitoring requests through the operation described in clause 5.2.1.1 below.

5.2.1.1 Continuous monitoring data stream API request

API description: This operation is used to request the P&P control instance to perform one shot monitoring operations periodically and put the results on specific websocket channels.

Direction: Vertical (through one-stop API) → Dedicated slice P&P control instance

Table 5 below provides attributes and parameters related to this API.

Table 5 – Continuous monitoring data stream API request attributes

Information element	Type	Mandatory/Optional/ Conditional	Description
ws_id	String	Mandatory	It is the identifier of the websocket channel.
element_id	String	Mandatory	It is the identifier of the element in the slice topological view offered by the P&P control instance to the vertical, and that exposes the target monitoring operation requested by the vertical.
api_id	String	Mandatory	It is the identifier of the target monitoring operation as expressed in the slice topological view offered by the P&P control instance to the vertical.

5.2.1.2 Continuous monitoring data stream API response

API description: When the vertical requests a continuous monitoring data stream through the API specified in clause 5.2.1.1 above, the P&P control instance opens a websocket channel. Since a single websocket can be opened per continuous monitoring session, and the vertical can request multiple data streams, the P&P augments each message in the data stream with appropriate attributes, in order to enable the consumer to distinguish between different data streams and monitoring information structures.

Direction: Dedicated per-slice P&P control instance → Vertical (via the one-stop API)

Table 6 below provides attributes and parameters related to this API.

Table 6 – Continuous monitoring data stream API response

Information element	Type	Mandatory/Optional/ Conditional	Description
slice_id	String	Mandatory	It is the identifier of the slice instance for which the vertical requested a monitoring a data stream.
owner_id	String	Mandatory	It is the identifier (in the P&P context) of the slice owner (i.e., an identifier for the vertical).
element_id	String	Mandatory	It is the identifier of the element in the slice topological view offered by the P&P control instance to the vertical, that was selected by the vertical in the monitoring data stream request.
api_id	String	Mandatory	It is the identifier of the target monitoring operation as selected by the vertical in the monitoring data stream request.
content	Object	Mandatory	It is the actual monitoring information and conveys the reply of the monitoring query for the "api_id" operation. The data follows the schema as defined in the slice topological view offered by the P&P control instance to the vertical, inline with the HATEOAS approach (see clause 5.2.2 below).

5.2.2 Runtime control APIs

On the other hand, the **runtime control APIs** represent one of the main innovative features of the P&P, as they are not bound to a specific pre-defined set of endpoints or resources. This is enabled by the way the P&P slice exposure information models are defined and conceived. In particular, the P&P framework follows an approach where the client and server sides of the REST API are decoupled and interact by means of dynamic hypermedia information provided by the server, similarly to the REST hypermedia as the engine of application state (HATEOAS) approach [b-REST]. In practice, the client side of the API (i.e., the vertical through the one-stop API) accesses the REST application (i.e., the P&P control instance) through a simple pre-defined endpoint offered by the server (which is used to retrieve the customized slice view). After that, all other interactions are based on information (i.e., endpoints and resource representations) that the client discovers directly from the server through dynamic hypermedia. In the case of the P&P control instance, this hypermedia is represented by the set of allowed APIs properties within each element in the slice view. This approach enables much more flexibility and extensibility compared with traditional service-oriented architectures where REST interactions happen over pre-defined and fixed interfaces shared through documentation or other out-of-band mechanisms. Therefore, it is not possible to derive in advance the full list of runtime control REST APIs that a given P&P control instance is able to expose. This also gives an unprecedented flavour of extensibility to the P&P framework, which can be very easily adapted to any vertical requirement in terms of expected runtime control granularity on the one hand, as well as in terms of slice exposure constraints set by slice providers.

5.3 Control plane

The main purpose of the control plane is to provide the slice control context through a set of configuration endpoints exposing technology and implementation-agnostic control APIs towards slice management and orchestration components.

The functionality is designed to handle tasks for the enforcement of network functions configuration rules and policies governing the runtime operations of radio access network (RAN), core network (CN), mobile edge computing (MEC), backhaul and wide area network (WAN) segments and it is composed of a set of components called control plane services (CPS) each offering specific configuration and control capabilities APIs as described below.

5.3.1 QoS control service

The QoS control service is responsible for enforcing dynamic QoS settings for each slice towards the underlying network segments (RAN, CN, MEC, Backhaul, WAN) according to the input parameters in the exposed interfaces.

The request to set the QoS constraints can be on a per-slice basis or for an IMSI and optionally for the EPS bearer IDs associated with a slice. In principle, the request to set the QoS constraints is deployed to all network segments depending on the slice composition

The QoSCPS exposes **APIs** for the following operations to its service consumers:

- Set QoS constraints, to allow indication of quantitative QoS parameters per slice to be enforced towards the required network segment.

Example usage:

[PUT | POST] ../qos/v1/qos-instance/{sliceId}/set_qos_constraints/{parameters}

All input parameters are described in Tables 7 and 8.

Table 7 – Set QoS constraints, parameters

Field name	Type	Example	Mandatory	Description
sliceId	UUID	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the slice associated to QOS.
segmentId	String	ACCESS	NO	Identify the type of the adapter, ACCESS or CORE adapter kind.
userEqId	String	49150123456789	NO	Identify the user equipment identity as IMSI.
epsBearerId	integer	5	NO	Identify the EPS bearer identity.

JSON parameters:

Table 8 – Set QoS constraints, JSON parameters

Field name	Type	Example	Mandatory	Description
qosDirName	String	UPL	YES	The setting directive for QoS parameters can be: UPL=Upload directive; DWL= Download directive.
qosDirValue	String	50	YES	The value of setting directive.
qosUnitScale	String	MB	YES	The measurement scale of the setting directive.

Request example:

In the following an example is reported for the setting of QoS parameter UPL (UpLink) to 50 MB given the <sliceID> UUID value, segmentId (ACCESS), userEqId (IMSI) and epsBearerId (5).

```
curl-X PUT http://localhost:8082/slicenet/ctrlplane/qos/v1/qos-
instance/<sliceID>/set_qos_constraints?segmentId=ACCESS&userEqId=49150123456789&epsBearerId=5-H
"Content-Type: application/json" -d'{"qosDirName":"UPL","qosDirValue":"50","qosUnitScale":"MB"}'
```

Success response: **200 OK**

Error response: **400 Bad Request**

- Set priority, to allow the change of priority for a traffic type flow associated to a slice within the back-haul data plane. Included parameters are:

Example usage:

[PUT] ../qos/v1/qos-instance/{sliceId}/set_priority/{parameters}

All input parameters are described in Table 9.

Table 9 – Set priority, parameters

Field name	Type	Example	Mandatory	Description
sliceId	UUID	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the slice associated to QoS.
priorityValue	Integer	1,….9	YES	Identify the priority value Identity.

Request example:

In the following, an example is reported for the setting of the priority parameter (1) given the <sliceID> UUID value.

```
curl-X PUT http://localhost:8082/slicenet/ctrlplane/qos/v1/qos-instance/<sliceID>/set_priority?priorityValue=1
```

Success response:

```
HTTP/1.1
200 OK
```

Error response:

```
HTTP/1.1
400 BadRequest
```

5.3.2 NF config control service

The NF config control service is responsible for enforcing the configuration operations towards the Network Functions composing the slice. Depending on the slice template/design that is instantiated a number of operations are expected to be available per slice. These operations are not limited to a predefined subset. Several operations can be defined and catalogued so that they can be included. The NF config control service analyses the requested operation and executes it in the context of a defined workflow among those configured during instantiation.

The NF config CPS exposes **APIs** for the following operations to its service consumers:

- Set NF configuration, to allow addition, update and removal of configuration as described in Table 10.

Table 10 – CONF-S supported operations

Endpoint	/function/{sliceid}/{operation}"
Request	POST/PUT/DELETE
Request body	A JSON array with configuration items as presented in the next table
Consumes	application/json
Produces	—-
Response body	—-
Return codes	200, 404 if the operation does not exist

The body of the request is structured as presented in Table 11.

Table 11 – Body request structure

Field	Type	Details
name	String	The name of one of the parameters of the operation. It is the same with the key defined in the operation descriptor.
value	String	The current value to be applied for the parameter under the above name.
unit	String	An optional unit identifier for numeric parameters.

5.3.3 Inter PoP connection control service (IPC)

The IPC control service is responsible, for each slice instance, for delivering a proper interconnection between the slice network functions (i.e., mostly VNFs and MEC applications) deployed in different segments and domains, namely edge (e.g., MEC) and core ones, allowing geographically distributed slice network functions to be properly interconnected according to their end-to-end forwarding graph descriptor.

The **IPC** CPS exposes **APIs** for the following operations to its service consumers:

- Provision InterPoP connections, to create a new interPoP connection among two or more PoPs where the network functions (i.e., VNFs and MEC applications) of a given slice have been deployed and provisioned.

This operation interconnects two network functions in a slice between PoPs within a single domain, through an interPoP connections descriptor representing the Backhaul SDN network between infrastructure pillars.

[PUT] ../ipc/v1/ipc-instance/{sliceId}/provision_interpop_connections

Example usage:

url: http://localhost:8081/slicenet/ctrlplane/ipc/v1/ipc-instance/{sliceId}/provision_interpop_connections

URL parameters:

operationId: provisionInterpopConnections

The interPoP connections object (Table 12) will contain the specific slice identifier and the list of interPoP paths to interconnect.

Table 12 – Provision interPoP connections parameters

Field name	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the slice associated with IPC.
InterPoP_Connections	array	[InterPoP_Path1, …, InterPoP_PathN]	YES	List of inter-PoP Paths.

Each interPoP_Path (Table 13) is an element of the list in the InterPoP connections, it contains a set of parameters: the endPoints to interconnect and the related constraints.

Table 13 – InterPoP path parameters

Field name	Type	Example	Mandatory	Description
endPoints	object	[endPointA, endPointB]	YES	A pair of routing devices on the PoPs to interconnect through the SDN backhaul network.
constraints	array	bandwidth=1GBPS	NO	Weights applied to a set of network resources, such as bandwidth, latency, etc.

The endPoints object (Table 14) contains the pair of IP addresses (either IPv4 or IPv6 format) and associated attributes of two nodes to interconnect.

Table 14 – endPoint parameters

Field name	Type	Example	Mandatory	Description
ipv4Address_nodeA/ ipv6Address_nodeA	string ipv4/ ipv6	"192.168.0.3"	YES	Source IP address either in IPv4 or IPv6 format.
ipv6Prefix_nodeA	string/ ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeA	string	"111"	YES	VLAN identifier of NodeA
ipv4Address_nodeB/ ipv6Address_nodeB	string ipv4/ ipv6	"192.168.12.4"	YES	Destination IP address either in IPv4 or IPv6 format.
ipv6Prefix_nodeB	string ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeB	string	"222"	YES	VLAN identifier of NodeB.

The *constraints* parameter (Table 15) contains the information about the bandwidth and latency to use along the path interconnecting the endPoints.

Table 15 – Constraints parameter

Field name	Type	Example	Mandatory	Description
bandwidth	float	1 GBPS	NO	Constraint that evaluates links based on available bandwidths.
latency	float	1 ms	NO	Constraint to keep latency below the specified value through a path.

Success response:

HTTP/1.1

200 OK

Error response:

HTTP/1.1

400 BadRequest

403 Forbidden

404 Not Found

500 Internal Server Error

- Update InterPoP connections, to allow dynamic modification of QoS constraints for the InterPoP Connections (e.g., bandwidth and latency).

[PUT] ../ipc/v1/ipc-instance/{sliceId}/update_interpop_connections

Example usage:

url: http://localhost:8081/slicenet/ctrlplane/ipc/v1/ipc-instance/{sliceId}/update_interpop_connections

URL parameters:

operationId: updateInterpopConnections

The interPoP connections object (Table 16) contains the specific slice identifier and the list of interPoP paths to update.

Table 16 – Update interPoP connections parameters

Field name	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the slice associated with IPC.
InterPoP_Connections	array	[InterPoP_Path1, …, InterPoP_PathN]	YES	List of inter-PoP Paths.

Each interPoP_Path (Table 17) is an element of the list in the InterPoP connections, it contains a set of parameters: the interconnected endPoints and the updated constraints to use.

Table 17 – InterPoP path parameters

Field name	Type	Example	Mandatory	Description
endPoints	object	[endPointA, endPointB]	YES	A pair of routing devices on the PoPs to interconnect through the SDN backhaul network.
constraints	array	bandwidth=1GBPS	NO	Weights applied to a set of network resources, such as bandwidth, latency, etc.

The endPoints object (Table 18) contains the pair of IP addresses (either IPv4 or IPv6 format) and associated attributes of two interconnected nodes.

Table 18 – endPoint parameters

Field name	Type	Example	Mandatory	Description
ipv4Address_nodeA/ ipv6Address_nodeA	string ipv4/ ipv6	"192.168.0.3"	YES	Source IP address either in IPv4 or IPv6 format.
ipv6Prefix_nodeA	string/ ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeA	string	"111"	YES	VLAN identifier of NodeA
ipv4Address_nodeB/ ipv6Address_nodeB	string ipv4/ ipv6	"192.168.12.4"	YES	Destination IP address either in IPv4 or IPv6 format.
ipv6Prefix_nodeB	string ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeB	string	"222"	YES	VLAN identifier of NodeB.

The *constraints* parameter (Table 19) contains the information about the bandwidth and latency to use along the path interconnecting the endPoints.

Table 19 – Constraints parameter

Field name	Type	Example	Mandatory	Description
bandwidth	float	2 GBPS	NO	Constraint that evaluates links based on available bandwidths.
latency	float	0.5 ms	NO	Constraint to keep latency below the specified value through a path.

Success response:

```
HTTP/1.1
200 OK
```

Error response:

```
HTTP/1.1
400 BadRequest
```

403 Forbidden
404 Not Found
500 Internal Server Error

- Remove InterPoP connections, this operation deletes the slice inter-PoP connections.

[DELETE] ../ipc/v1/ipc-instance/{sliceId}/remove _interpop_connections

Example usage:

url: http://localhost:8081/slicenet/ctrlplane/ipc/v1/ipc-instance/{sliceId}/remove _interpop_connections

URL parameters:

operationId: removeInterpopConnections

The interPoP connections object (Table 20) contains the specific slice identifier and the list of interPoP paths to disconnect.

Table 20 – Remove InterPoP connections parameters

Field name	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the slice associated with IPC.
InterPoP_Connections	array	[InterPoP_Path1, …, InterPoP_PathN]	YES	List of inter-PoP paths to disconnect.

Each interPoP_Path (Table 21) to remove contains the endPoints to disconnect.

Table 21 – InterPoP Path parameters

Field name	Type	Example	Mandatory	Description
endPoints	object	[endPointA, endPointB]	YES	A pair of routing devices on the PoPs to be interconnected through the SDN backhaul network.

The endPoints object (Table 22) contains the pair of IP addresses (either IPv4 or IPv6 format) and associated attributes of two nodes to disconnect.

Table 22 – endPoint parameters

Field name	Type	Example	Mandatory	Description
ipv4Address_nodeA/ ipv6Address_nodeA	string ipv4/ipv6	"192.168.0.3"	YES	Source IP address either in IPv4 or IPv6 format.
ipv6Prefix_nodeA	string/ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeA	string	"111"	YES	VLAN identifier of NodeA.
ipv4Address_nodeB/ ipv6Address_nodeB	string ipv4/ipv6	"192.168.12.4"	YES	Destination IP address either in IPv4 or IPv6 format.

Table 22 – endPoint parameters

Field name	Type	Example	Mandatory	Description
ipv6Prefix_nodeB	string/ipv6	prefix/length in bits	NO	The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets.
vlan_nodeB	string	"222"	YES	VLAN identifier of NodeB.

Success response:

```
HTTP/1.1
200 OK
```

Error response:

```
HTTP/1.1
400 BadRequest
403 Forbidden
404 Not Found
500 Internal Server Error
```

6 eHealth use case

Figure 5 shows the scenarios of the eHealth UC where the ambulance is connected to the hospital with a 5G network. There are two services running in this use case:

- the BlueEye service and
- the Telestroke assessment service.

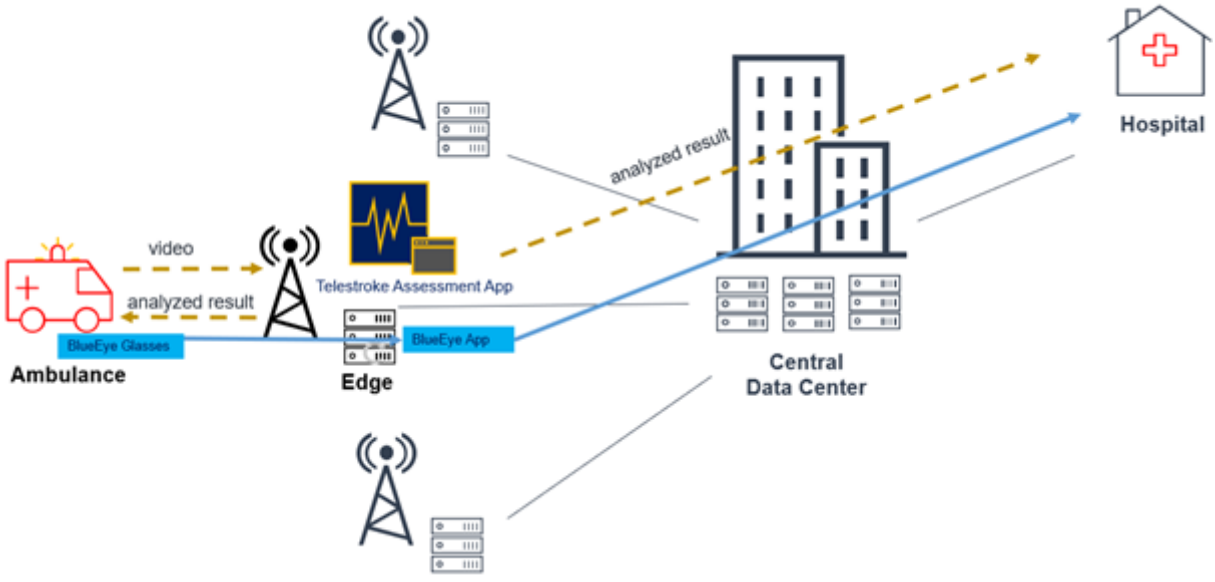

Figure 5 – eHealth UC high level description

The BlueEye service is initiated with the BlueEye glasses worn by a paramedic in the ambulance capturing a live video of the scene. The glasses connect to the BlueEye app running at the edge for authentication and also to set up the communication with the hospital. The first version of the BlueEye

App provides a direct link from the BlueEye glasses to the hospital where the doctors can view video of the scene.

The Telestroke assessment service consists of the gateway app running in the ambulance, capturing the video of the patient, and the Telestroke assessment app running at the edge, collecting the data from the gateway and runnings a ML algorithm to analyse the images for the stroke condition. The assessment app will send the analysed result back to the paramedic and also to the hospital for expert to examine.

Figure 6 shows a SliceNet business model, as applied in the eHealth use case.

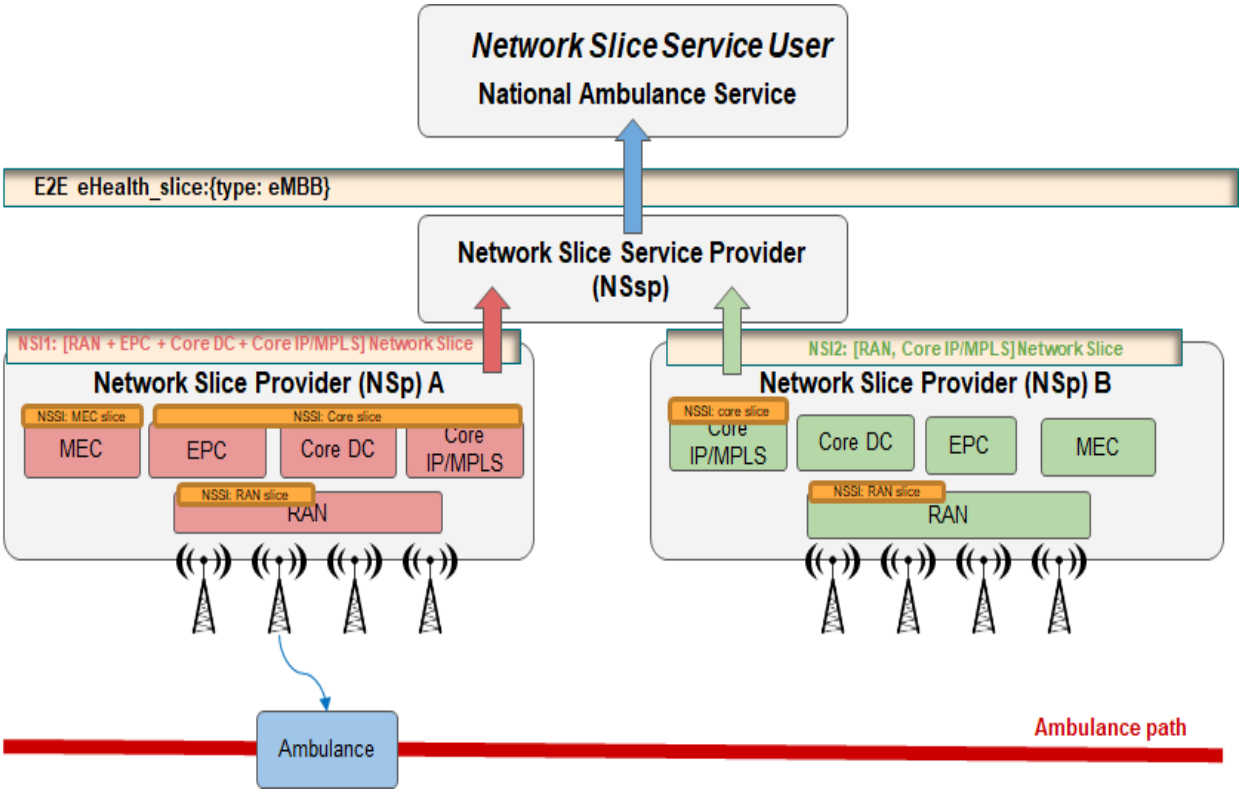

Figure 6 – eHealth Business Model with NSsu /service provider perspective

The end-to-end (E2E) slice customer is a national/regional health service organization (e.g., national ambulance service in Ireland), which operates multiple (static) hospitals, dispatch centres, and (moving) ambulances. The E2E eHealth slice offered to the customer by the network service slice provider (NSsp) consists initially of a "base" network slice instance (NSI) containing the minimal set of network functions and services. The base slice is fairly static and is centred on a geographical area in the vicinity of the hospital. The hospital/dispatch centre hosts experts who provide real-time support to the paramedics. As ambulances are dispatched, additional network sub-slice instances (NSSIs) may be instantiated in order to increase the geographical coverage of the slice (e.g., by adding RAN) or to guarantee the latency and availability requirements of the slice. For the latter, additional processing functions may be dynamically instantiated at suitable MEC locations. Dispatch may trigger a handover of a paramedic's communication stream to a different hospital. Handover between domains might be needed while the ambulance is moving.

7 Conclusions

This document proposes a new framework that enables vertical QoE-aware network slice management empowered by machine learning technologies. The framework can derive QoE level indications for vertical businesses from QoS parameters and allow feedback from vertical businesses

to a network slicing system. Such vertical feedback and QoS-QoE mapping will then be explored to inform configuration actions or other actuations to be executed so that the network slice belonging to that vertical is modified accordingly to meet the dynamic requirements of the vertical. The overall framework is presented with a set of essential components described including one-stop API to allow vertical feedback, plug and play control to customize the vertical's view and control, an overlay control plane on top of the 4G/5G control plane and data plane to enforce the actions/actuations, and a cognition sub-plane that is in charge of QoE-QoS mapping. An eHealth use case is employed to illustrate the key processes in this framework.

Bibliography

[b-ITU-T Y.3174] Recommendation ITU-T Y.3174 (2020), *Framework for data handling to enable machine learning in future networks including IMT-2020.*

[b-ONAP] Open Network Automation Platform (ONAP), https://docs.onap.org/projects/onap-modeling-modelspec/en/latest/

[b-REST] REST HATEOAS approach, https://en.wikipedia.org/wiki/HATEOAS

[b-WebSocket] I. Fette, A. Melnikov et al. (2011), *The WebSocket Protocol, RFC 6455*. https://tools.ietf.org/html/rfc6455