

Draft new Recommendation ITU-T Y.3183 (ex.Y.ML-IMT2020-VNS)

Framework for network slicing management assisted by machine learning leveraging QoE feedback from verticals

Summary

This Recommendation provides a framework for machine learning assisted network slicing management, leveraging vertical end users' feedback on QoE, which can help achieve run-time optimisation of user perceived performance. The overall architecture, components, workflow and related APIs of this framework are specified with respect to the high-level requirements identified. A use case is provided in appendix to show an application example of this framework. Example implementations of the key APIs are also provided.

Keywords

ML, network slice, QoE, QoS, vertical feedback

Table of Contents

1.	Scope.....	4
2.	References.....	4
3.	Definitions	5
3.1	Terms defined elsewhere:.....	5
3.2	Terms defined in this Recommendation.....	6
4.	Abbreviations and acronyms	6
5.	Conventions	7
6.	Introduction.....	8
7.	High-level requirements	9
7.1	General requirements.....	9
7.2	Functional requirements	10
8.	Framework and workflow.....	11
8.1	Overall framework.....	11
8.2	QoE feedback workflow	13
9.	Components and related APIs	16
9.1	OSA	16
9.2	P&P Controller	17
9.2.1	Continuous monitoring APIs	17
9.2.2	Runtime control APIs	18
9.3	Control Plane	18
9.3.1	QoS Control Plane Service	19
9.3.2	NF Config Control Plane Service	19
9.3.3	Inter PoP Connection Control Plane Service.....	19
10.	Security considerations	20
	Appendix I. eHealth Use Case	21
	Appendix II. Examples of P&P Controller APIs	23
A.	Example of continuous monitoring data stream API request	23
B.	Continuous monitoring data stream API response	23

C.	QoE feedback API	24
	Appendix III. Examples of Control plane APIs	25
A.	Example usage of QoS Control	25
B.	Example usage of NF Config Control	27
C.	Example usage of IPC Control	28
	Bibliography.....	34

Draft new Recommendation ITU-T Y.3183 (ex.Y.ML-IMT2020-VNS)

Framework for network slicing management assisted by machine learning leveraging QoE feedback from verticals

1. Scope

This Recommendation provides a framework for machine learning-assisted network slicing leveraging verticals' feedback to improve the performance. It addresses the following subjects:

- Overview and requirements;
- Framework for machine learning assisted network slice (NS) management leveraging input from verticals, focusing on QoE feedback;
- Components and APIs in the framework.

An eHealth use case is provided in Appendix to show an application example of this framework.

2. References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T X.1601] Recommendation ITU-T X.1601 (2015), *Security framework for cloud computing*.
- [ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), *Security requirements for NGN release 1*.
- [ITU-T Y.3104] Recommendation ITU T Y.3104 (2018), *Architecture of the IMT-2020 network*.
- [ITU-T Y.3110] Recommendation ITU T Y.3110 (2017), *IMT-2020 network management and orchestration requirements*.
- [ITU-T Y.3111] Recommendation ITU T Y.3111 (2017), *IMT-2020 network management and orchestration framework*.
- [ITU-T Y.3170] Recommendation ITU-T Y.3170 (2018), *Requirements for machine learning-based quality of service assurance for the IMT-2020 network*.
- [ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.
- [ITU-T Y.3174] Recommendation ITU-T Y.3174 (2020), *Framework for data handling to enable machine learning in future networks including IMT-2020*.
- [ITU-T Y.3177] Recommendation ITU-T Y.3177 (2021), *Architectural framework for artificial intelligence-based network automation for resource and fault management in future networks including IMT-2020*.
- [ITU-T Y.3178] Recommendation ITU-T Y.3178 (2021), *Functional framework of AI-based network service provisioning in future networks including IMT-2020*.

3. Definitions

3.1 Terms defined elsewhere:

This Recommendation uses the following terms defined elsewhere:

3.1.1 control plane [b-ITU-T Y.2011]: The set of functions that controls the operation of entities in the stratum or layer under consideration, plus the functions required to support this control.

3.1.2 data plane [b-ITU-T Y.2011]: The set of functions used to transfer data in the stratum or layer under consideration.

NOTE – In the ITU-T IMT-2020 related standard documents, "User plane" is used preferentially rather than "Data plane".

3.1.3 management [b-ITU-T Y.3100]: In the context of IMT-2020, the processes aiming at fulfilment, assurance, and billing of services, network functions, and resources in both physical and virtual infrastructure including compute, storage, and network resources.

3.1.4 network function [b-ITU-T Y.3100]: In the context of IMT-2020, a processing function in a network.

NOTE 1 – Network functions include but are not limited to network node functionalities, e.g., session management, mobility management and transport functions, whose functional behaviour and interfaces are defined.

NOTE 2 – Network functions can be implemented on a dedicated hardware or as virtualized software functions.

NOTE 3 – Network functions are not regarded as resources, but rather any network functions can be instantiated using the resources.

3.1.5 network infrastructure management provider [b-ITU-T Y.3103]: The network infrastructure management provider integrates infrastructures of multiple infrastructure providers to offer the combined resources to the NS management and orchestration provider.

3.1.6 network infrastructure provider [b-ITU-T Y.3103]: The network infrastructure provider is the owner, the provider and the manager of the network infrastructure.

3.1.7 network slice [b-ITU-T Y.3100]: A logical network that provides specific network capabilities and network characteristics.

NOTE 1 – Network slices enable the creation of customized networks to provide flexible solutions for different market scenarios which have diverse requirements, with respect to functionalities, performance and resource allocation.

NOTE 2 – A network slice may have the ability to expose its capabilities.

NOTE 3 – The behaviour of a network slice is realized via network slice instance(s).

3.1.8 network slice blueprint [b-ITU-T Y.3100]: A complete description of the structure, configuration and work flows on how to create and control a network slice instance during its life cycle.

NOTE – A network slice template can be used synonymously with a network slice blueprint.

3.1.9 network slice instance [b-ITU-T Y.3100]: An instance of network slice, which is created based on a network slice blueprint.

NOTE 1 – A network slice instance is composed of a set of managed run-time network functions, and physical/logical/virtual resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the service instance(s).

NOTE 2 – A network slice instance may also be shared across multiple service instances provided by the network operator. A network slice instance may be composed of none, one or more sub-network slice instances which may be shared with another network slice instance.

3.1.10 network slice management and orchestration provider [b-ITU-T Y.3103]: The NS management and orchestration provider is responsible for orchestrating NS(s) and managing the lifecycle of NS instance(s) based on NS blueprint(s) (a term defined in [b-ITU-T Y.3100] (see definition in clause 3.1.2)).

NOTE – The NS blueprints can be provided by third parties.

3.1.11 network slice provider [b-ITU-T Y.3103]: The NS provider is the owner of the NS instance(s) and provides the NS instance(s).

3.1.12 network slice service provider [b-ITU-T Y.3103]: The NS service provider is the user of the NS instance(s), and is responsible for providing services to its NS service users via the NS instance(s).

3.1.13 network slice service user [b-ITU-T Y.3103]: The NS service user uses the service(s) provided by the NS instance(s).

3.1.14 network sub-slice management and orchestration provider [b-ITU-T Y.3103]: The network sub-slice management and orchestration provider is responsible for orchestrating network sub-slice(s) and managing the lifecycle of the network sub-slice instance(s).

3.1.15 network sub-slice provider [b-ITU-T Y.3103]: The network sub-slice provider is the owner and provider of the network sub-slice instance(s).

3.1.16 orchestration [b-ITU-T Y.3100]: In the context of IMT-2020, the processes aiming at the automated arrangement, coordination, instantiation and use of network functions and resources for both physical and virtual infrastructures by optimization criteria.

3.1.17 quality of experience (QoE) [b-ITU-T P.10]: The degree of delight or annoyance of the user of an application or service.

3.1.18 quality of service (QoS) [b-ITU-T Q.1741.9]: The collective effect of service performances which determine the degree of satisfaction of a user of a service. It is characterized by the combined aspects of performance factors applicable to all services, such as:

- service operability performance;
- service accessibility performance;
- service retainability performance;
- service integrity performance; and
- other factors specific to service.

3.1.19 role [b-ITU-T Y.3502]: A set of activities that serves a common purpose.

3.1.20 user plane (UP) [b-ITU-T Y.1714]: Refers to the set of traffic forwarding components through which traffic flows.

NOTE – "User plane" is referred to as "transport plane" in other ITU-T Recommendations.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

None.

4. Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Programming Interface
AI	Artificial Intelligence
CP	Control Plane

CPS	Control Plane Services
CPSR	Control Plane Service Register
CSP	Customer Services Provider
DP	Data Plane
DPP	Data Plane Programmability
E2E	end-to-end
IMT-2020	International Mobile Telecommunications-2020
ML	Machine Learning
NF	Network Function
NImp	Network Infrastructure management provider
NS	Network Slice
NSS	Network Sub-Slice
NSI	Network Slice Instance
NSsu	Network Slice service user
NSsp	Network Slice service provider
NSp	Network Slice provider
NIp	Network Infrastructure provider
NSmop	Network Slice management and orchestration provider
NSSp	Network Sub-Slice provider
NSSmop	Network Sub-Slice management and orchestration provider
OSA	One-Stop Access
P&P	Plug and Play
PoP	Point of Presence
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
SLA	Service Level Agreement
UE	User equipment
VNF	Virtual Network Function

5. Conventions

In this Recommendation:

The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.

The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option, and the feature can be optionally enabled by the network

operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.

6. Introduction

Network slicing is a primary approach in future networks including IMT-2020 and beyond to meet the diverging services' requirements from diverse business verticals by creating multiple logical networks with required Quality of Service (QoS). Network slicing connects virtual and/or physical Network Functions (NFs) [b-ITU-T Y.3100] together with the required underlying resources through chaining to create the intended service. Truly end-to-end network slicing needs to create properly isolated logical networks across all the physical network segments all the way from source to destination, including Radio Access Network (RAN), Mobile/Multi-access Edge Computing (MEC), Backhaul/Transport Network, Core Network (CN), and Wide Area Network (WAN), regardless of the heterogeneity of the underlying infrastructures. Moreover, there are various aspects to be addressed to achieve advanced network slicing, including guaranteed QoS and customized network services, among others. Guaranteed QoS can be achieved through QoS control of the various network segments via radio resource allocation specific to various and often heterogeneous wireless/mobile systems and data plane programmability for diverse wired data path technologies. Customised network services are created through Virtual Network Function (VNF) or more generally Network Function (NF) configuration and chaining across the interested Points of Presence (PoPs). Therefore, a set of NS control services is necessitated.

On top of these network slicing capabilities, for optimised network slicing operation, Machine Learning (ML) can be used to enhance network slicing performance. Figure 1 illustrates this vision.

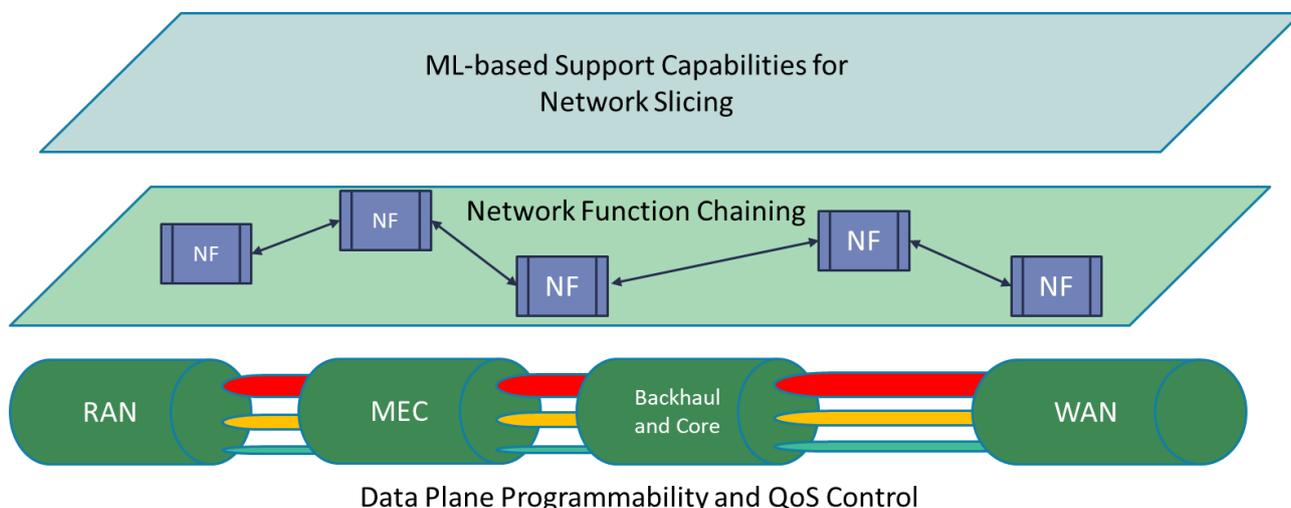


Figure 1 ML-based support for network slicing

Furthermore, it is highly desirable that verticals are allowed to customize their NS-based services and provide as inputs the customization requirements in not only the service provisioning phase but also the runtime phase. Upon receiving such vertical feedback, the management system of a future network should apply the feedback to modify the service, and the corresponding end-to-end (E2E) NS for that vertical. Depending on the level of the vertical's feedback, the management system may need to convert Quality of Experience (QoE) requirements into QoS constraints and/or Service Level Agreement (SLA) terms. Subsequently, these QoS/SLA constraints are employed to influence the modification of the NS instance and/or inform other related operations to support the fulfilment of the requirements in terms of QoS/SLA constraints. For autonomous network control and management, the above steps are required to be automated, enabled by ML, automation and other related technologies.

Such quality-aware NS control is a challenging task that requires a proper framework and tools in place to be executed efficiently. Of particular importance is the quality perceived by the network slice service users (NSsu) [b-ITU-T Y.3103], especially the verticals, since it has a correlation with their degree of satisfaction with the deployed service. As such, it becomes essential to capture the verticals' perspective about the runtime behaviour of the deployed NS and its influence on the services deployed on top. Network slice service providers (NSsp) [b-ITU-T Y.3103] will usually have contractual relationships with multiple types of verticals (e.g., automotive industry, power distribution, eHealth), and thus they will require a mechanism for gauging the satisfaction of the vertical customers, i.e., their QoE with the deployed NS.

The NSsp cooperate with the network slice providers (NSp) [b-ITU-T Y.3103] that create the corresponding NS in order to provide the required services to the NSsu.

Moreover, an inclusive stakeholder interface is expected for a common approach for various stakeholders to support all their interactions with the network slicing system via different projections of the information accessible for different purposes, based on the role of the stakeholder who consumes or influences the information and even operations. This will allow the stakeholders to utilise wireless technologies of various generations such as IMT-2020 and beyond, support dynamicity and adaptability with respect to the varied and distributed demands of the served verticals, allow for the expression of a multitude of performance, security and other requirements, enable various stakeholders to be able to participate as providers, and allow verticals to have certain level of control over their NSs for applying the principles of their application logic.

Furthermore, the delivery of heterogeneous services customized for a large set of vertical businesses with diverging requirements (e.g., in terms of QoS/SLA and QoE) requires an effective and common framework able to keep verticals in the loop of services and NSs provisioning, from design to operation. However, current offerings from service providers and network operators are still based on a very limited involvement of verticals and consumers in the runtime control and management of the offered services. This gap needs to be filled along the development of network slicing and the stakeholder interface, by allowing a vertical to have extended control capabilities of their NS-based services in a controlled manner. In practice, dynamic exposure control is the key enabler for this evolution of services and NS control exposure towards a paradigm where verticals play an active role in the runtime control and management of their NSs.

This Recommendation describes a framework that enables vertical QoE-aware NS control empowered by ML technologies. This framework allows QoE feedback from vertical businesses to facilitate a network slicing system to optimise the runtime slicing and thus the corresponding verticals' NS-based services. Such vertical feedback will then be explored to inform configuration actions or other actuations to be executed so that the NS 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 a stakeholder interface to allow the vertical feedback (and other stakeholders to access the system), dynamic exposure control to customize the vertical's view and control with continuous monitoring capabilities, and an overlay control plane on top of IMT-2020 and beyond control plane and data plane to enforce the actions/actuations. A set of associated Application Programming Interfaces (APIs) for monitoring or control are described (with implementation details included in Appendixes II and III). An eHealth use case is used to illustrate the key processes in this framework (more details are presented in Appendix I). Finally, it is noted that there is no limitation of the framework to other scenarios of applicability beyond network slicing.

7. High-level requirements

7.1 General requirements

REQ-G1. The framework is required to allow verticals to provide runtime QoE feedback, and map verticals' QoE feedback to NS QoS reconfiguration.

NOTE 1 – Quality perceived by the NSsu [b-ITU-T Y.3103] needs to be measured in runtime, and represented in an interoperable format, agnostic to NSsp specific implementation. A standard mapping of this QoE feedback to specific NS QoS metrics, such as latency or bitrate is needed.

REQ-G2. The framework is required to have a common, aggregated stakeholder interface for various stakeholders to access network slicing services.

NOTE 2 – Support for dynamicity and adaptability with respect to the varied and distributed demands of the served verticals, together with other involved business roles, is needed.

REQ-G3. The framework is required to have dynamic exposure control to allow exposing network slicing capabilities to verticals.

NOTE 3 – Verticals may need to have certain level of control over their NSs for applying the principles of their application logic. Mechanisms such as NS templates which can be customized using ML can be used.

REQ-G4. The framework is required to have a set of NS control services to address the various control requirements for advanced E2E network slicing.

REQ-G5. The framework is required to facilitate the integration of ML techniques for intelligent services.

7.2 Functional requirements

REQ-F1. A common mechanism is required to express verticals' QoE feedback in a machine-readable format regardless of the different expectations from the different verticals and the multiple service semantics.

NOTE – For instance, the Mean Opinion Score [b-ITU-T P.800.1] can be employed for visual communication services.

REQ-F2. A continuous monitoring mechanism is required to allow a NSp [b-ITU-T Y.3103] to monitor the quality of deployed NSs as perceived by its NSsu, independently of the particularities of the NS infrastructure and the service supported on top of it.

REQ-F3. An API is required to allow the execution of the feedback mechanism independently from the NSsp specific implementation and the endpoints exposed to the NSsu.

REQ-F4. A QoE-to-QoS mapping mechanism is required to allow translating a vertical's feedback into specific configurable NS QoS metrics.

REQ-F5. The QoE-to-QoS mapping mechanism is required to be able to combine gathered QoE feedback with monitored NS QoS metrics to finally derive under what QoE conditions which NS parameters have to be re-configured, enabling the possibility of mixing and matching monitoring and analysis functions in an autonomous workflow.

REQ-F6. A mechanism is required to allow a built-in approach for onboarding new services to benefit from ML models in the ML pipeline in order to achieve intelligent operations such as QoE optimisation.

REQ-F7. A plug-in-based approach for new service onboarding is recommended for the ease of deployment and intelligence extensibility.

REQ-F8. A stakeholder access interface is required to accommodate the diversity of NSsu needs, using a flexible approach with respect to the way various stakeholders interact dynamically to offer an adequate synthesis aiming at catering for those needs.

REQ-F9: A dynamic approach is required so that the interacting stakeholders are identified based on a producer/consumer relationship in order to allow role-based access to the services offered by the framework.

REQ-F10. The stakeholder access interface is required to display different projections of the onboarded information (descriptors) based on identified business roles.

REQ-F11. Exposure control is required to expose well-defined monitoring, management and/or control capabilities in line with the stakeholders' different roles to allow the stakeholders especially the verticals to be equipped with the corresponding facilities in order to optimise their own NS-based service.

REQ-F12. The exposure control, together with the associated API, is required to allow the stakeholders especially the verticals to have management access to their NSs and apply tailored control logics on top of them through a set of customized control operations.

REQ-F13. NS control services are required to achieve the network slicing capabilities including network function chaining and configuration, and QoS control over the data plane.

REQ-F14. APIs are required for the network slicing control services to operate over heterogeneous network segments of heterogeneous networking technologies.

8. Framework and workflow

8.1 Overall framework

To meet the listed requirements in Clause 7, a framework is defined as shown in Figure 2 Figure 3.

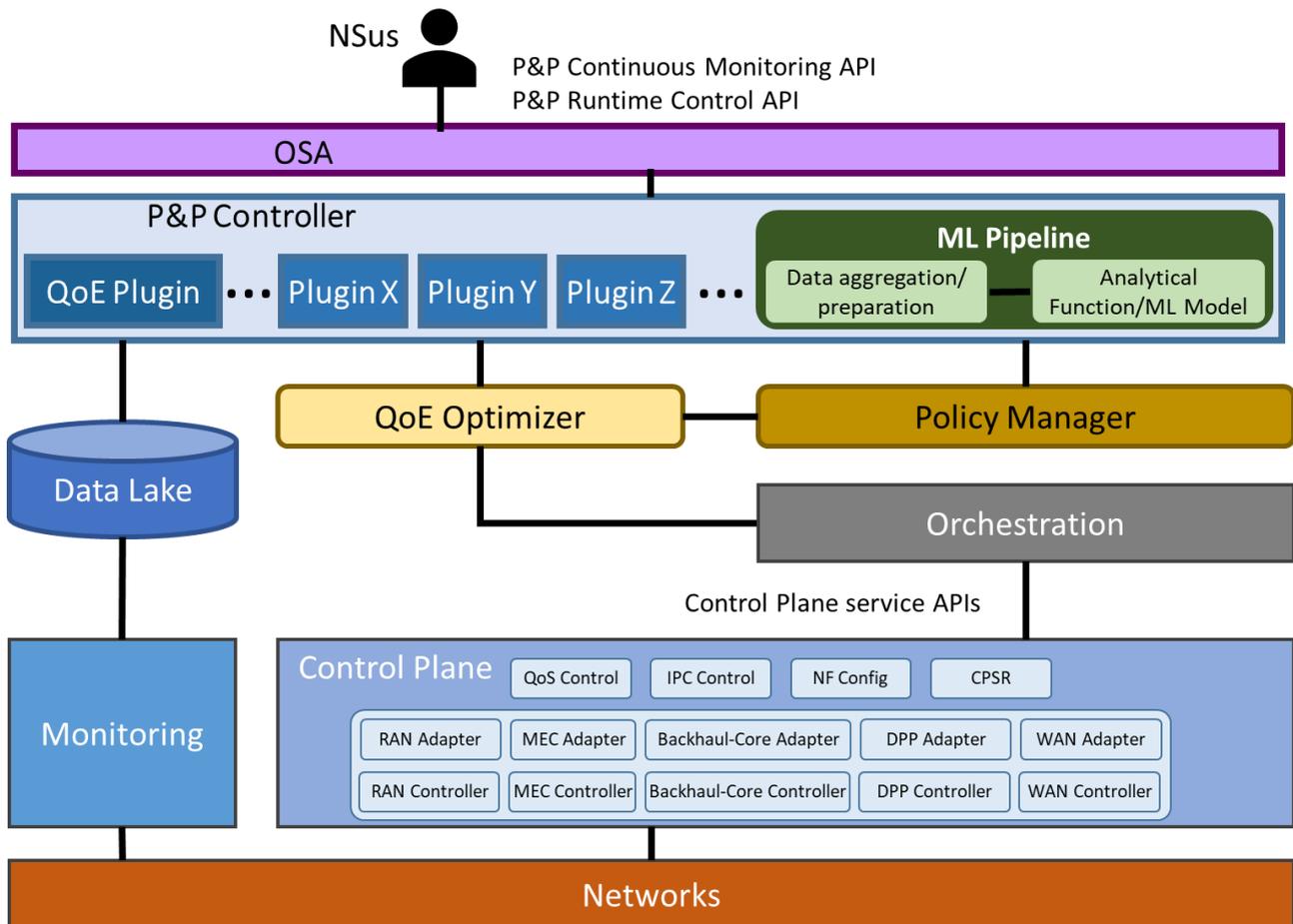


Figure 2 Overall framework

The framework meets all the general requirements (**REQ-G1 to G5**). The framework consists of the following components that satisfy the functional requirements (**REQ-F1 to F14**):

- The One-Stop Access (OSA) component offers a stakeholder interface and service access mechanisms that provide a common access platform to utilize the services provisioned by the framework for all the stakeholders including vertical customers (**REQ-F8, F9**), supporting role-based customized frontend view (**REQ-F10, F11**), and facilitates the collection of QoE feedback into the framework (**REQ-F1**).

NOTE 1 – Further details are provided in Clause 9.1.

- The Plug & Play (P&P) (Exposure) Controller component exposes continuous service monitoring and runtime ML-based management and control capabilities to NSsus (**REQ-F12, F13**), and it allows onboarding new P&P services, e.g., to enhance ML capabilities (**REQ-F6, F7**). The ML pipeline, comprising data aggregation/preparation and analytical function and the ML model, processes the vertical's QoE feedback data and the network QoS metrics to provide insights that enable the ML-based actuation. Analysis is applied first to historical data to learn new models and derive new policies and optimization parameters. At run-time, similar analysis is applied to verify and refine the learned models. The learned models are inferred in near-real-time to provide concrete input for actuation.

NOTE 2 – Further details are provided in Clause 9.2.

- The control plane component with a set of control services configures or reconfigures NSs over heterogeneous network segments, and enforces different aspects of network slicing including QoS Control, Inter-PoP Connection Control (IPC Control) for service chaining, Network Function Configuration (NF Config.), and a service registration component Control Plane Service Register (CPSR). In addition, a set of controllers and adaptors are in place to enforce the specific actuation actions required and deal with the heterogeneity of the underlying network segments including RAN, MEC, Backhaul network, CN, WAN and the Data Plane Programmability (DPP, e.g., based on [b-NSDPP]**Error! Reference source not found.**), respectively (**REQ-F14**).

NOTE 3 – Further details are provided in Clause 9.3.

The rest of the components deal with management capabilities and include the following:

- The Monitoring component collects QoS metrics related to the concerned NS instances.
- The Data Lake component stores and persists the monitored data from the Monitoring component. Its implementation can be, e.g., based on the Data Lake as defined in [b-DL] (**REQ-F2, F3**).
- The Policy Manager component governs the actuation policies based on the output of the ML pipeline. Its implementation, e.g., can be based on the ONAP policy framework [b-ONAP].
- The QoE Optimizer component triggers the desired vertical QoE feedback informed actuations based on the policies stored in it and updated by the Policy Manager.
- The Orchestration component provides network orchestration capabilities at the various levels including Network Function Virtualisation (NFV), E2E service, and the underlying resources. Its implementation can be, e.g., based on Open Source MANO [b-ETSI OSM].

The workflow described in Clause 8.2 addresses **REQ-F4, F5**.

NOTE 4 – In the framework described above, all the components are aware of NSs: the Monitoring and the Data Lake components collect and store per-NS QoS measurements, the P&P Controller component provides per-NS control and per-NS view to verticals via the OSA component, the Policy Manager and QoE Optimizer components update and maintain per-NS policies, and the Orchestration component outputs per-NS actuation action to the control plane component, which offers a set of control services specifically related to NS operations. Meanwhile, the components of this framework can be generalized to support general network control and management operations, e.g., the Monitoring component can collect QoS measurements related to traffic that does not belong to any NS, and the control plane component, organized according to a service-oriented architecture, can

onboard non-NS related control services. Therefore, this framework, in principle, can be applied to generic ML-assisted network control and management scenarios.

The framework and its components are in line with those defined in the relevant existing ITU Recommendations for network slicing such as IMT-2020 network management and orchestration [ITU-T Y.3110] [ITU-T Y.3111], architectural framework for ML in IMT-2020 [ITU-T Y.3172] [ITU-T Y.3177] [ITU-T Y.3178], and ML-based QoS assurance [ITU-T Y.3170]. In particular, the framework is in line with the functional model of machine learning-based QoS assurance for the IMT-2020 network defined in [ITU-T Y.3170], in terms of overall logic and components including QoS monitoring, QoS pre-processing, QoS policy decision, and interactions with management and data planes, and it complements the functional model by adding OSA and QoE-QoS mapping for vertical feedback, P&P control including APIs for QoE plugin, overlay control plane services and APIs for network slicing over heterogeneous network segments. The framework is also compatible with high-level architectural components defined in [ITU-T Y.3172] in terms of the ML pipeline.

8.2 QoE feedback workflow

The framework described in clause 8.1 allows runtime network slicing control enabled by ML and informed by QoE feedback from NSsus (verticals). In the following, the procedure steps of the QoE feedback workflow shown in Figure 3 are explained.

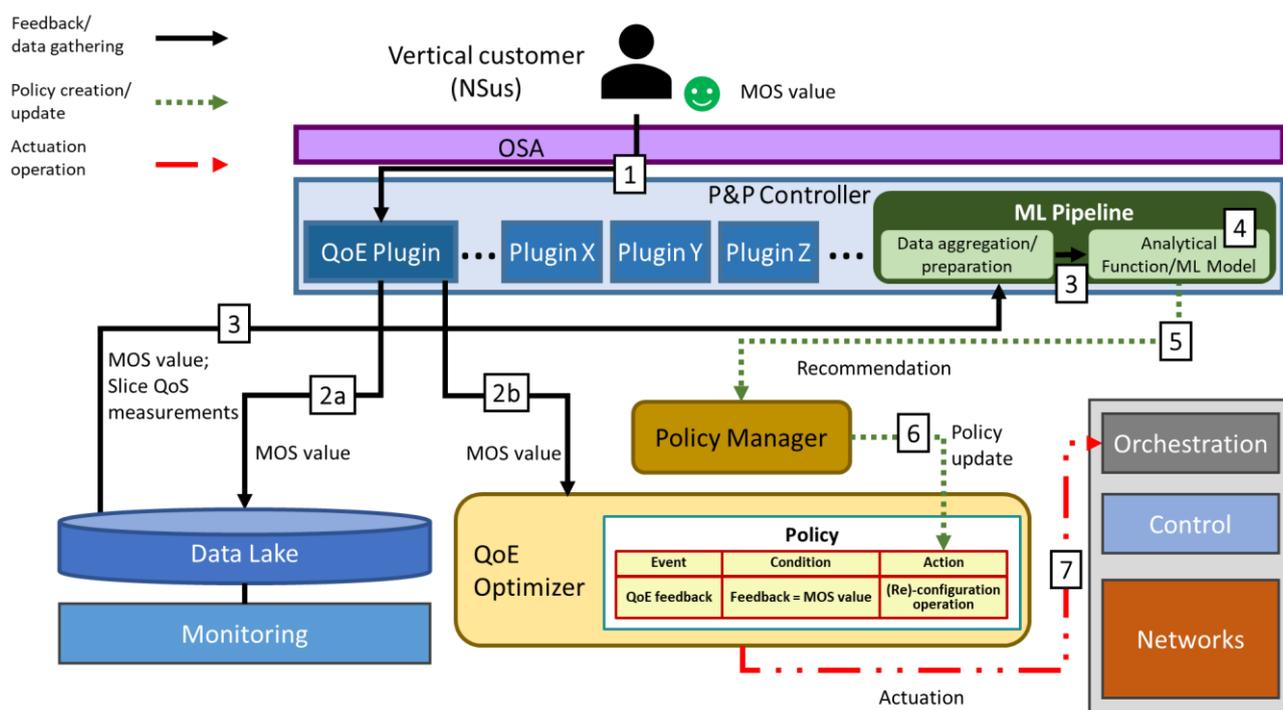


Figure 3 QoE feedback workflow

Thanks to the capabilities exposed through the OSA component, which grants a vertical customer access to the underlying NS control and management capabilities, the QoE feedback mechanism allows collecting a qualitative or quantitative rating – e.g., a MOS value – from the NSsu through a dedicated function enclosed within a specialized plugin (QoE plugin) in the P&P Controller component (Step 1 in Figure 3).

The collected rating can then be captured by the QoE Optimizer component (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 a NSsu feedback, the policy would state that, given a specific MOS value (e.g., “Bad”), the QoE Optimizer component should trigger the

specified (re-)configuration operation towards the corresponding enforcement point for the operation (i.e., the NSsp's orchestration system) to maintain optimal quality levels. This policy is initially distributed from the Policy Manager component upon the instantiation of the NS and all software artefacts responsible for its control and management. The parameters of the policy (i.e., the ECA items) are configured according to the requirements of the NS and the service it needs to support.

Such a policy system decouples the “what” from the “how” of the defined actions: the QoE Optimizer component is the entity responsible to know the type of action that needs to be enforced under certain stimulus (the event), but it does not have the knowledge on the capacity for the execution of the several operations required to materialize the action (i.e., the “how”). This capacity is assumed by the enforcement point, such as the Orchestration component, 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 QoE from the NSsu perspective, it does not necessarily indicate which underlying parameters of the configured NS have a greater influence on the QoE. In fact, the feedback provides an opinion view about the service provided, while the monitoring of the underlying NSs and network is focused on measurable QoS metrics, 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 NS or network side (i.e., the QoS).

Then, thanks to QoE/QoS mapping, it is possible to identify the NS and network parameters that can be re-configured to influence on the quality as it is perceived by the NSsu. Indeed, by setting up policies which tie the vertical's feedback with an action in turn which (re-)configures the most relevant QoS parameters that are affecting the expressed QoE, it is possible to fully realize the “verticals-in-the-loop” vision that IMT-2020 is advocating.

The inclusion of verticals in the control loops requires that feedback options are expressed in the context of the vertical businesses. This in turn is because NSsp's resources are offered in a technology abstraction manner that hides underlying complexity and exposes only a number of properties from a predefined set onto which the underlying capabilities can be mapped. Through this way, the verticals can see a set of available options that are relevant to their businesses without knowledge of the mechanisms that realise them. Such an option allows also different vendors to provide their enabling artifacts on conditions that adhere to the predefined set of defined options (e.g., sensing, actuation and optimisation) so that, once selected, dependencies can be resolved to allow the artifacts to operate as expected.

To fully satisfy the requirements, the framework provides the proper analysis of the received QoE feedback to influence the policy system and the related actions in order to apply the most suitable (re-)configuration according to the current NS status. Starting from the QoE plugin, which has certain dependencies on input types, the collected MOS value is inserted into the Data Lake (Step 2a).

Upon insertion in the Data Lake component, a specialized analytical function/ML model collects the QoE values and the QoS measurements of the concerned NS during the NS subscription (Step 3) in a periodic manner in order to perform background analysis of both types of information. The ML models are located within an ML pipeline, in which the data aggregation/preparation modules pre-process the data to make it ready for consumption by the ML model in Step 3.

Thanks to the capabilities of the P&P Controller component, the ML pipeline and the ML model can be plugged, along with other plugins, such as the QoE plugin, to perform specific analysis related to the type of NS that is deployed and the service supported over it. For the particular case of the QoE feedback mechanism, the monitoring dependency is resolved from the high-level requirements and selected as options by the vertical. In this resolution process based on the involved administrative domains, the abstracted technology artifacts are engaged for the NS and their monitoring output is made available as metric types that the higher-level components can process. The goal of the ML

model is to analyse the expressed feedback and the monitored QoS at the NS level, to identify the main cause(s) for unsatisfactory quality levels (Step 4).

After this analysis, the model then recommends to the Policy Manager component which QoS parameter(s) to be (re)configured as part of policy actions (Step 5). Given these gained insights, the Policy Manager component then updates the Action field of the policies related to the expressed vertical feedback for NS quality maintenance, i.e., the QoE Optimizer component (Step 6).

With this updated policy, the QoE Optimizer component triggers the most suitable action according to the most updated insights, following the same procedure as in Step 7.

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

Table 1 Vertical QoE feedback mechanism

Step	Description	Additional details
1	Thanks to the capabilities exposed through the QoE plugin, the vertical customer expresses its feedback about the quality of the experience.	The feedback takes form, for instance, as a MOS value, e.g.: <ul style="list-style-type: none"> • Scalar: 1, 2, 3, 4 or 5 • String: Very Bad, Bad, Regular, Good, Very Good
2a	The feedback is inserted into the Data Lake component for its later consumption by ML pipelines.	A specific schema in the Data Lake component 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 NS for which the feedback has been collected.
2b	The feedback is directly consumed by the QoE Optimizer component for immediate actuation to maintain quality levels.	As a pre-condition, a policy should be already instantiated at the QoE Optimizer component specifying the ECA values for the actuation related to the feedback.
3	The feedback from the NSsu and the NS QoS parameters are gathered from the Data Lake component by an ML pipeline to perform their analysis and derive insights.	The QoS parameters are aggregated and prepared per end-to-end NS by the Monitoring component. Then, pre-processing modules within the ML pipeline further prepare the data to be ML ready, such as normalization and smoothing of the data. As such, the ML model consumes the data from the data aggregation/preparation modules within the pipeline, which has already gathered from the schemas of the Data Lake component the information tagged under the identifier of the NS to which the ML model belongs to.
4	The ML model, as a pluggable function of the NS control, performs the analysis of the consumed data.	The ML model is provided by an ML provider, e.g., the same NS provider or the vertical customer. The P&P Controller component only provides a platform that allows the dynamic instantiation of pluggable control functions per NS.
5	After the analysis, the ML model contacts the Policy Manager component to recommend which are the QoS parameters of the underlying NS that need to be (re-)configured in case undesirable QoE situations are reported through the feedback mechanism.	The recommendation from the ML model should indicate which of the analysed QoS parameters of the NS gathered from the Data Lake need to be stated as to be re-configured 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 Manager

		component, a common information model for the action should be in place, for example, as defined in [ITU-T Y.3174].
6	The Policy Manager component disseminates the updated policy about the treatment of the feedback from the NSSu to the QoE Optimizer component in order to have the most up-to-date action.	The Policy Manager component updates the QoE optimizer component, e.g., by publishing and subscription. Such mechanisms would enable gathering the latest policies related to the specific NS identifier to which the policy refers to.
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 with a concrete (re-)configuration to be carried out, the QoE Optimizer component contacts the Orchestration component to apply the action.	The QoE Optimizer component is only responsible of determining the ‘when’ and ‘what’ of the actuation, while the ‘how’ is the responsibility of the Orchestration component, which should engage with the control plane component to achieve the specified (re-)configurations with the specified parameters.

9. Components and related APIs

9.1 OSA

The service-based architecture approach followed by the IMT-2020 core specification [ITU-T Y.3104] is paving the way for a fully modular and softwarized approach of network services.

The OSA component provides role-based access to the services supported by the framework to the different business roles, i.e., NSp, NSsp and NSSu [b-ITU-T Y.3103].

The following OSA operations are foreseen per role:

- NSp
 - actuation descriptors onboarding
 - monitoring descriptors onboarding
 - resource type registration associated with actuation and monitoring descriptors
 - identification of resource types availabilities
 - onboarding of policy-based rules with dependencies on actuation and monitoring descriptors
 - identification of network coverage
 - resources list based on type and capabilities
- NSsp
 - technology agnostic information of actuation and/or monitoring descriptors
 - identification of monitoring and actuation capabilities offered by NSp
 - selection of SLA related automation rules
 - indication of the monitoring and actuation options which should be activated per NS type
 - registration of QoE plugin and P&P services
 - registration of NS templates with dependencies on QoE and P&P services
 - NS templates list based on vertical-tailored services and underlying dependencies including network coverage per service
 - NS templates annotation with service features on QoE and P&P
 - NSSu information of NS templates annotated with service features
 - selection of service features to be activated

- access to NS specific actuation and monitoring information as exposed via P&P services

9.2 P&P Controller

The P&P Controller component aims at providing truly customized runtime control, management and operation of E2E NS instances while offering vertical-tailored services. It enables NS customization, as it provides a novel combination of tailored control functions and APIs that allow verticals to customize their NS instances according to their needs. In practice, the P&P Controller component provides a control environment, dedicated per NS, which offers to the verticals direct control of NS instances and provides a flavour of customization. The main idea is that the P&P Controller component provides, for each NS instance, a vertical-tailored abstraction of the NS (including its view and the available control services) on top of the APIs exposed by the control plane component.

The vertical-tailored services are constituted by the combination of:

- A NS abstraction, which offers an abstract and vertical friendly view, on top of which the logic for the control operations is added so that these control operations can be consumed by verticals at runtime over their NSs. The runtime consumption is done by means of a dynamic and automatic mapping of operations issued from the vertical-tailored APIs into specific actions to be enforced or performed through the pluggable control functions.
- Vertical-tailored APIs, which provide the set of tailored control and management APIs that are exposed to the vertical and offer the runtime control operations over the NS-specific view. The vertical accesses these APIs through the OSA component.

The P&P vertical-tailored APIs rely on a generalized NS exposure that enables a vertical customized abstraction of NS instances, still agnostic to the network slicing technology and able to address the description of managed network entities from vertical service, to NS, and down to network functions and resources. This generalized exposure provides a NS topology view that is augmented with a set of runtime control operations that are offered to customize the NS control, management and monitoring: these operations are in practice the customized control primitives at disposal of the vertical. The P&P vertical-tailored APIs can be defined as the APIs (e.g., implemented as REST APIs) that a given P&P control instance exposes to the vertical for enabling tailored runtime control. These APIs can be grouped into two categories:

- **Continuous monitoring APIs** – to expose towards the vertical continuous data streams including tailored NS performance metrics and statistics.
- **Runtime control APIs** – to expose towards the vertical available control services.

9.2.1 Continuous monitoring APIs

The **continuous monitoring APIs** provide access to NS monitoring information and can be considered as a basic control exposure feature allowing verticals to be aware of statistics and performances related to their NS.

The nature of the monitoring data accessible is subject to the P&P control customization process, and all kinds of resources, network functions, NS or service-level monitoring information can be in principle exposed to verticals according to the agreed control exposure. The P&P Controller component exposes NS related monitoring information as continuous streams of data.

This approach transforms single-query monitoring operations offered usually through an HTTP GET in a periodically executed query that pushes results on a proper channel, creating a data stream. The technology used to create such a channel can be based on, e.g., the concept of WebSocket [b-IETF WS]. Each P&P control instance is able to handle one or more channel in order to meet the requirement of having multiple streams per NS. Once the channel is opened, a vertical can request through the OSA the continuous monitoring of desired performance metric (e.g., through a graphical user interface), on the condition that such metric can be retrieved by a GET runtime control operation already exposed by the given P&P control instance. In practice, a vertical can request the P&P control

instance to periodically execute a given monitoring query and then collect the results through the opened channel as continuous data streams. The vertical, through the OSA component, can perform such continuous monitoring requests through the operation described below.

9.2.1.1 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 channels.

Direction: Vertical (through the OSA component) → dedicated per-NS P&P control instance. An example of attributes and parameters related to this API can be found in Appendix II.

9.2.1.2 Continuous monitoring data stream API response

API description: When the vertical requests for a continuous monitoring data stream through the continuous monitoring data stream API request, the P&P control instance opens a channel. Since a single channel can be opened per continuous monitoring session, and the vertical can request multiple data streams, the P&P control instance augments each message in the data stream with proper attributes, in order to make the consumer able to distinguish different data streams and monitoring information structures.

Direction: Dedicated per-NS P&P control instance → Vertical (through the OSA component). An example of attributes and parameters related to this API can be found in Appendix II.

9.2.2 Runtime control APIs

The **runtime control APIs** allow verticals to have a degree of control over the NS owned by them. The P&P Controller component can follow an approach where client and server sides of the REST API are decoupled and interact via dynamic hypermedia information provided by the server, similarly to the REST Hypermedia As The Engine Of Application State (HATEOAS) approach [b-HATEOAS]. In practice, the client side of the API (i.e., the vertical through the OSA component) 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 NS 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 a P&P control instance, this hypermedia is represented by the set of allowed APIs' properties within each element in the NS view. This approach enables flexibility and extensibility when 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. This also gives extensibility to the P&P Controller component, which can be adapted to any vertical requirements in terms of expected runtime control granularity, and in terms of NS exposure constraints set by NS providers.

In the case of a vertical's QoE feedback, the specific actuation exposed to the vertical is to enable the vertical to report QoE feedback through the OSA component.

An example of attributes and parameters related to this API can be found in Appendix II.

9.3 Control Plane

The main purpose of the control plane component is to provide NS control services through a set of configuration endpoints that expose control APIs towards the management and orchestration components (as shown in Figure 2). The control plane handles tasks for the enforcement of network functions configuration rules and policies governing the runtime operations of the various network

segments, and offers a set of Control Plane Services (CPS), each supporting specific configuration and control capabilities' APIs as described in the following sub-clauses.

9.3.1 QoS Control Plane Service

The QoS Control Plane Service (QoS CPS) is responsible to enforce dynamic QoS setting for each NS towards the underlying network segments according to the input parameters in the exposed interfaces between the control plane component and the Orchestration component. The request to set the QoS constraints can be on a per-NS basis or for an International Mobile Subscriber Identity (IMSI), among others. The request to set the QoS constraints is deployed to all network segments depending on the NS composition.

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

- Set QoS Constraints, to allow indication of quantitative QoS parameters per NS to be enforced towards the target network segment.
- Set Priority, to allow change of priority for a traffic type flow associated to a NS within the data plane.

An example usage of the QoS CPS can be found in Appendix III.

9.3.2 NF Config Control Plane Service

The NF Config Control Plane Service (NF Config CPS) is responsible to enforce the configuration operations towards the network functions composing a NS. Depending on the NS template that is instantiated, a number of operations are expected to be available per NS. These operations are not limited to a predefined subset. The NF Config CPS analyses the requested operation and executes it in the context of a defined workflow among the operations configured during the NS instantiation.

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

- Set NF configuration, to allow addition, update and removal of configurations.

An example usage of the NF Config CPS can be found in Appendix III.

9.3.3 Inter PoP Connection Control Plane Service

The IPC Control Plane Service (IPC CPS) is responsible, for each NS, to deliver a proper interconnection of the network functions deployed in the different network segments, allowing geographically distributed network functions to be properly interconnected according to their E2E forwarding graph descriptor.

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

- Provision InterPoP Connections, to create a new interPoP connection among two or more PoPs where the network functions of a given NS have been deployed and provisioned.
This operation interconnects two network functions in a NS between PoPs, through an interPoP connections descriptor representing the backbone network between the PoPs.
- Update InterPoP Connections, to allow dynamic modification of QoS constraints for the InterPoP connections (e.g., bandwidth, latency).
- Remove InterPoP Connections, to delete the NS inter-PoP connections.

An example usage of the IPC CPS can be found in Appendix III.

10. Security considerations

This Recommendation presents the architectural framework for machine learning assisted network slicing management leveraging QoE feedback from verticals, which is expected to be applied in future networks including IMT-2020.

General network security requirements and mechanisms in future networks are applicable [ITU-T X.1601] [ITU-T Y.2701] [ITU-T Y.3101].

It is required to prevent from unauthorized access to, and data leaking from, the framework functionalities, especially to the OSA component functionalities by the different stakeholders (the OSA component providing the northbound interface of the framework functionalities), and to the ML pipelines. Implementation of mechanisms regarding authentication and authorization, and external attack protection should be considered.

Appendix I. eHealth Use Case



Figure A. eHealth use case high-level description

Figure shows the scenarios of the eHealth UC where the ambulance is connected to the hospital with an IMT-2020 network. There are two services running in this use case, including a video communication service via wearable video glasses and a ML-based stroke diagnosis assessment service. The video communication service is initiated with the glasses equipped with a video camera and worn by a paramedic in the ambulance capturing a live video of the scene. The glasses connect to the corresponding video communication App running at the edge for authentication to set up the communication with the hospital, providing a direct link from the glasses to the hospital where the doctors can view video of the scene. The stroke assessment service consists of the Gateway app running in the ambulance, capturing the video of the patient, and the stroke assessment app running at the edge, collecting the data from the Gateway and run a ML algorithm to analyse the images for the stroke condition. The assessment app will send the analysed result back to the paramedic and to the hospital for expert to examine.

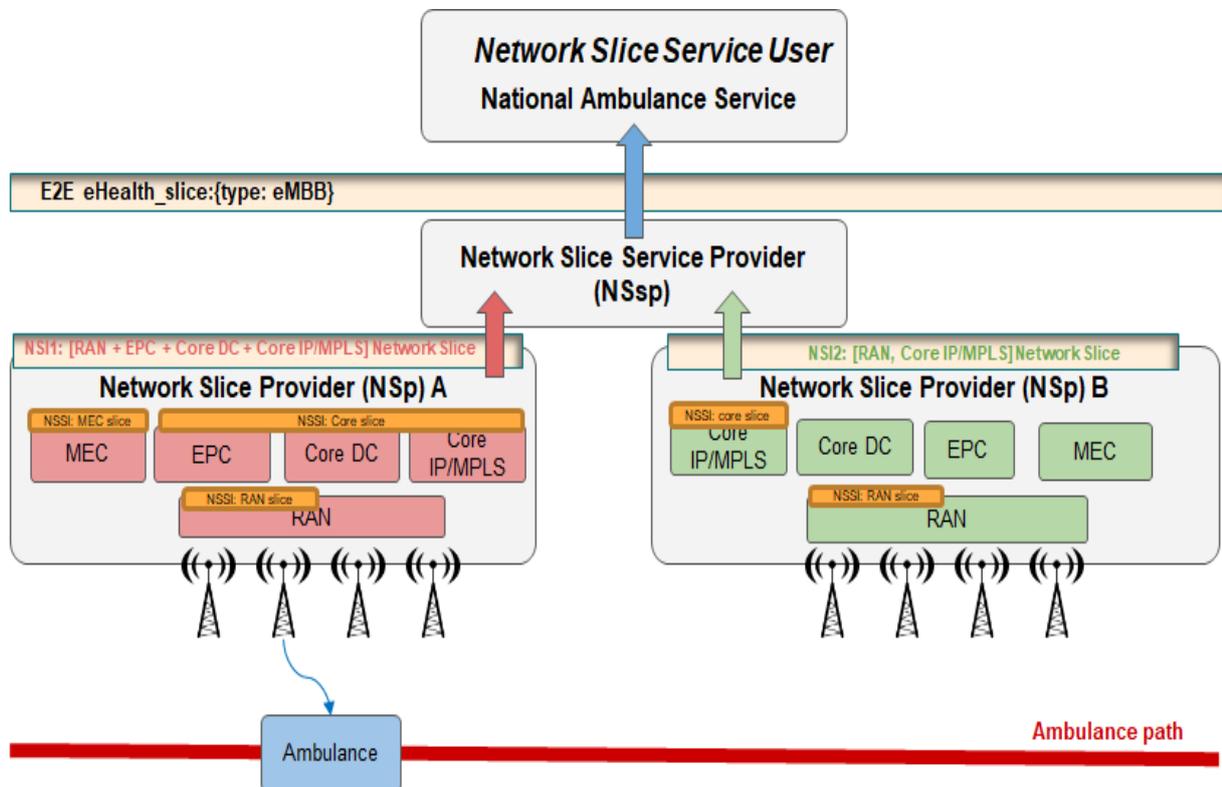


Figure B eHealth Business Model with NSsu and NSsp perspectives

Figure shows a business model, as applied in the eHealth use case. The E2E NS customer is a national/regional health service organization, which operates multiple (static) hospitals, dispatch

centres, and (moving) ambulances. The E2E eHealth NS offered to the customer by the NSsp consists initially of a “base” network slice instance (NSI) containing the minimal set of network functions & services. The base NS is static and is centred on a geographical area in the vicinity of the hospital. The hospital/dispatch hosts experts who provide real-time support to the paramedics. As ambulances are dispatched, additional network sub-slice instances (NSSIs) may be instantiated to increase the geographical coverage of the NS (e.g., by adding RAN) or to guarantee the latency and availability requirements of the NS. 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.

Appendix II. Examples of P&P Controller APIs

A. Example of continuous monitoring data stream API request

Table A provides attributes and parameters related to this API.

Table A. 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 NS 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 NS topological view offered by the P&P control instance to the vertical.

B. Continuous monitoring data stream API response

Table B provides attributes and parameters related to this API.

Table B. Continuous monitoring data stream API response attributes

Information Element	Type	Mandatory/Optional /Conditional	Description
slice_id	String	Mandatory	It is the identifier of the NS instance for which the vertical requested a monitoring data streams.
owner_id	String	Mandatory	It is the identifier (in the P&P context) of the NS owner (i.e., an identifier for the vertical).
element_id	String	Mandatory	It is the identifier of the element in the NS 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 NS topological view offered by the P&P

			control instance to the vertical, in line with the HATEOAS approach.
--	--	--	--

C. QoE feedback API

Table C provides attributes and parameters related to this API.

Table C. QoE feedback API attributes

Information Element	Type	Mandatory/Optional /Conditional	Description
slice_id	String	Mandatory	It is the identifier of the NS instance for which the vertical requested a monitoring data streams.
owner_id	String	Mandatory	It is the identifier (in the P&P context) of the NS owner (i.e., an identifier for the vertical).
qoe_value	String	Mandatory	It is the QoE value input by the vertical.

Appendix III. Examples of Control plane APIs

A. Example usage of QoS Control

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

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

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

The Set QoS constraints, parameters are described in the Table D.

Table D. Set QoS constraints, parameters

Information Element	Type	Example	Mandatory	Description
sliceId	String	6b699c12-d154-49ed-8922- f578e108f818	YES	Unique ID of the NS 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.

The JSON parameters are described in Table E.

Table E. Set QoS constraints, JSON parameters

Information Element	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> String 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&epsBe  
arerId=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 change of priority for a traffic type flow associated to a NS within the back-haul data plane. Included parameters are:

Example usage:

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

The Set Priority parameters are described in Table F.

Table F. Set Priority parameters

Information Element	Type	Example	Mandatory	Description
sliceId	String	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the NS associated to QoS.
priorityValue	Integer	1, 2, 3...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> String 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

B. Example usage of NF Config Control

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 G.

Table G. NF Config supported operations

Endpoint	/function/{sliceid}/{operation}"
Request	POST/PUT/DELETE
Request Body	A JSON array with configuration items as presented in Table H.
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 H.

Table H. NF Config 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.

C. Example usage of IPC Control

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 NS have been deployed and provisioned.

This operation interconnects two network functions in a NS between PoPs intra 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 contains the specific NS identifier and the list of interPoP paths to interconnect.

The Provision interPoP Connections parameters are described in Table I.

Table I. Provision interPoP Connections parameters

Information Element	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the NS associated to IPC.
InterPoP_Connections	array	[InterPoP_Path1, ..., InterPoP_PathN]	YES	List of inter-PoP Paths.

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

The InterPoP Path parameters are described in Table J.

Table J. InterPoP Path parameters

Information Element	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 and latency.

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

The endPoint parameters are described in Table K.

Table K. endPoint parameters

Information Element	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 ipv4Address_nodeB/ ipv6Address_nodeB	string string ipv4/ ipv6	"111" "192.168.12.4"	YES YES	VLAN identifier of NodeA 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 contains the information about the bandwidth, latency to use along the path interconnecting the endPoints.

The constraints parameters are described in Table L.

Table L. constraints parameters

Information Element	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 under specified latency 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, 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 contains the specific NS identifier and the list of interPoP paths to update.

The Update interPoP Connections parameters are described in Table M.

Table M. Update interPoP Connections parameters

Information Element	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the NS associated to IPC.
InterPoP_Connections	array	[InterPoP_Path1, ..., InterPoP_PathN]	YES	List of inter-PoP Paths.

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

The InterPoP Path parameters are described in Table N.

Table N. InterPoP Path parameters

Information Element	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 and latency.

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

The endPoint parameters are described in Table O.

Table O. endPoint parameters

Information Element	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 contains the information about the bandwidth, latency to use along the path interconnecting the endPoints.

The constraints parameters are described in Table P.

Table P. constraints parameter

Information Element	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 under specified latency 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 NS 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

Table Q contains the specific NS identifier and the list of interPoP paths to disconnect.

Table Q. InterPoP Connections parameters

Information Element	Type	Example	Mandatory	Description
sliceId	string	6b699c12-d154-49ed-8922-f578e108f818	YES	Unique ID of the NS associated to IPC.
InterPoP_Connections	array	[InterPoP_Path1, ..., InterPoP_PathN]	YES	List of inter-PoP Paths to disconnect.

Table R contains the endPoints to disconnect.

Table R. InterPoP Path parameters

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

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

Table S. endPoint parameters

Information Element	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.

Success response:

HTTP/1.1
200 OK

Error response:

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

Bibliography

- [b-ITU-T P.10] Recommendation ITU-T P.10/G.100 (2017), *Vocabulary for performance, quality of service and quality of experience*.
- [b-ITU-T P.800.1] Recommendation ITU-T P.800.1 (2016), *Mean opinion score (MOS) terminology*.
- [b-ITU-T Q.1741.9] Recommendation ITU-T Q.1741.9 (2015), *IMT-2000 references to Release 11 of GSM evolved UMTS core network*.
- [b-ITU-T Y.1714] Recommendation ITU-T Y.1714 (2009), *MPLS management and OAM framework*.
- [b-ITU-T Y.2011] Recommendation ITU-T Y. 2011 (2004), *General principles and general reference model for Next Generation Networks*.
- [b-ITU-T Y.3100] Recommendation ITU-T Y.3100 (2017), *Terms and definitions for IMT-2020 network*.
- [b-ITU-T Y.3103] Recommendation ITU-T Y.3103 (2018), *Business role-based models in IMT-2020*.
- [b-ITU-T Y.3502] Recommendation ITU-T Y.3502 (2014), *Information technology – Cloud computing –Reference architecture*.
- [b-ETSI OSM] ETSI Open Source MANO (OSM): <https://osm.etsi.org/>.
- [b-IETF WS] I. Fette, A. Melnikov, “The WebSocket Protocol”, IETF RFC 6455, December 2011, <https://tools.ietf.org/html/rfc6455>.
- [b-NSDPP] R. Ricart-Sancheza, P. Malagonb, A. Matencio-Escolar, J. M. Alcaraz Calero, and Q. Wang, “Towards Hardware-Accelerated QoS-aware 5G Network Slicing Based on Data Plane Programmability”, (Wiley) Transactions on Emerging Telecommunications Technologies, Vol. 31, No. 1, 2020.
- [b-DL] N. Miloslavskaya, A. Tolstoy, “Big Data, Fast Data and Data Lake Concepts”, Elsevier Procedia Computer Science, vol. 88, pp. 300-305, October 2016.
- [b-ONAP] Open Network Automation Platform webpage: <https://docs.onap.org/en/latest/>.
- [b-HATEOAS] J. Thijssen, “What is HATEOAS and why is it important for my REST API?”, REST CookBook, October 2016.
-