Deliverable D5.2

Modelling and Design of Vertical-Informed QoE Sensors

Editor(s):	Joanna Balcerzak, Orange Labs Networks (France)
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public
Contractual delivery date:	30 th November 2018
Actual delivery date:	15th December 2018
Suggested readers:	Infrastructure Providers, Communication Service Providers, Digital Service Providers, Network Operators, Digital Service Customers, Vertical Industries, Telecommunication/ICT Professionals
Version:	1.0
Total number of pages:	78
Keywords:	QoE, QoS, Sensor, Network Slice, Multi-Domain, 5G, Stakeholders, Technical Use-Cases, Architecture

Abstract

This document reports the activities related to the design of QoE Sensors. It is the first deliverable to be submitted within the WP5 activities. To ensure the coherency and fluidity of the coming WP5 deliverables, the current D5.2 details the design of Slicing QoE sensor while taking into account the QoE metrics defined within the three project use cases (Smart Grid, Smart City, eHealth). In this regards, the design aspects for a monitoring framework for raw and QoS and the QoE monitoring architecture were studied and elaborated.

QoE sensors are responsible for generating QoE metrics using the collected QoS values. This can be assured through a specific function that defines the relationship between QoS and QoE or using a predictive model for QoE based on classification/estimation that learns through training. Both options are depicted. QoE shall be understood to be a multi-

dimensional concept which ranges over different aspects of quality and how users perceive it.

The code of the noisy neighbour model will be published on GitHub or gitlab when WP8 platform is available. The code details the training and prediction phases for anomalies detection model implementation with R language will be provided in GitHub.

Disclaimer

The research leading to these results has received funding from the European Union Horizon 2020 Programme under grant agreement number H2020-ICT-2014-2/761913.

Impressum

[Full project title] End-to-End Cognitive Network Slicing and Slice Management Framework inVirtualised Multi-Domain, Multi-Tenant 5G Networks

[Short project title] SLICENET

[Number and title of work-package] WP5– Cognitive, Service-Level QoE Management

[Number and title of task] T5.2 - Modelling, Design and Implementation of Vertical-Informed QoE Sensors

[Document title] Modelling and Design of Vertical-Informed QoE Sensors

[Editor: Name, company] Joanna Balcerzak, Orange Labs Networks (France)

[Work-package leader: Name, company] Dean Lorenz, IBM Israel

Copyright notice

Executive summary

QoE is a subjective measure that involves human dimensions. It links the user's perception, expectations, and experience with application and network performance. However, measuring human perception is a complex task. In the context of slicing, measuring perceived QoE is a challenging problem for verticals because it is costly and complex due to the human involvement in the process. The challenging issue of the subjective measurement is to predict it from the objective measurements; in other words predict QoE from a given set of QoS parameters. QoE shall be understood to be a multi-dimensional concept which ranges over different aspects of quality and how users perceive it and it not only depends on the technical performance of the transmission and delivery chain but also on a wide range of other factors, including content, application, user expectations and goals, and context of use From the users' point of view, the quality of a service depends as well on their physical characteristics and on their socio-economic and cultural background. The influence of a given factor on the way the quality of service is assessed by a user may be quite different depending on the actual situation.

The deliverable D5.2 focuses on the design of the QoE sensors. It is the first deliverable to be submitted within the WP5 activities. To ensure the coherency and fluidity of the coming WP5 deliverables, the current D5.2 details the design of Slicing QoE sensor while taking into account the QoE metrics defined within the three project use cases (Smart Grid, Smart City, eHealth). In this regards, the design aspects for a monitoring framework for raw and QoS and the QoE monitoring architecture were studied and elaborated. The latter includes the main functions leading to interpret and map raw data to the expected QoE metrics. Specifically, the current deliverable recall QoE and QoS for the networking slicing context, define the QoE and QoS for each use case and the mapping and aggregations between them.

In addition and towards end to end cognitive network slicing, in this deliverable we propose to use machine learning techniques and the automated underlying pipeline (data preparation, pre-processing, etc.) to predict the QoE metrics from raw and QoS data. This is framed as "Intelligent Sensor". To exemplify the approach, two illustrations are presented: Intelligent sensor for Anomaly Detection and Intelligent sensor for Noisy Neighbour prediction.

The implementation of the QoE Sensor depends on the QoE metrics, the identification of slice resources and their exposed QoS metrics. Since that data is under collection by the time of the deliverable submission, the two illustrations of "the intelligent Sensor" have been evaluated using real network data extracted from internal networks of Orange France.

Based on this QoE sensor design, metadata requirements, and the machine learning approach for the intelligent sensor, the implementation will be followed within the D5.3 and within the WP6 for the monitoring part.

List of authors

Company	Authors	Contributions
Orange France	Joanna Balcerzak Marouane Mechteri Yosra Ben Slimen Ruby Krishnaswamy Ying Ren	Editor of the document; Elaboration of the inputs; definition of KPI for the 3 project use cases; defining the problem statement for the eHealth(Multi-handover); Design of the QoE Sensor; definition of key machine learning aspects for QoE sensors; Sensor for anomaly forecasting and noisy neighbour prediction
Orange Romania	Marius Iordache Catalin Brezeanu Mihai Idu	Smart City use case owner and description of QoS and QoE metrics.
Altice Labs	Pedro Miguel Neves	Smart Grid use case owner and description of QoS and QoE metrics.
Eurecom	Navid Nikaein, Xenofon Vasilakos	Monitoring framework design aspects; Participation in the discussions about QoE sensor definition and the problem statement for the eHealth (Multi-handover)
University of the West of Scotland		UWS will report contribution regarding sensing in D6.2.
IBM ISRAEL-Science and technology LTD	Dean H. Lorenz	Monitoring in Support of Cognitive Slice Management section

List of reviewers

Company	Reviewer
Dell EMC	Thuy Truong
Universitat Politècnica de Catalunya-BarcelonaTech	Salvatore Spadaro
Eurescom	Anastasius Gavras
Orange France	Bertrand Decocq Imen Grida Ben Yahia

Table of Contents

List of figures

List of tables

Abbreviations

5G	Fifth Generation (mobile/cellular networks)
5G PPP	5G Infrastructure Public Private Partnership
API	Application Program Interface
BER	Bit Error Rate
CP	Control Plane
FDA	Functional Data Analysis
FED	Federal Reserve System
FPCA	Functional Principal Component Analysis
HoN	Health of Network
IoT	Internet of Things
KPI	Key Performance Indicator
M2M	Machine to Machine
MF	Monitoring Framework
ML	Machine Learning
PCA	Principal Component Analysis
QoE	Quality of Experience
QoI	Quality of Information
QoS	Quality of Service
R&D	Research and Development
SDN	Software Defined Networks
SLICENET	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualized Multi-Domain, Multi-Tenant 5G Networks
UE	User Equipment
UP	User Plane
NSP	Network Service Provider
DSP	Distributed Service Provider
NSR	Network Slice Resource
E2E	End to End
IMSI	International Mobile Subscriber Identity
SAP	Service Access Point
TTI	Transmission Time Interval
VM	Virtual Machine
VNF	Virtual Network Function

1 Introduction

The main objective of this deliverable is to design the QoE sensors for multi-domain eHealth and Smart Grid use-cases, as well as for a single domain Smart City use-case.

QoE shall be understood to be a multi-dimensional concept which ranges over different aspects of quality and how users perceive it. From the users' point of view, the quality of a service depends as well on their physical characteristics and on their socio-economic and cultural background. The influence of a given factor on the way the quality of service is assessed by a user may be quite different depending on the actual situation.

The scope of this deliverable is to

- Defining QoE and QoS for the networking slicing context from the latest state of the art and beyond.
- Detailing the QoE and QoS for each use case.
- Elaborating the mapping and aggregations between the QoE and QoS metrics from each use case.
- Designing a QoE sensor with respect to the corresponding metadata and data model
- Proposing a novel approach of Intelligent QoE Sensor based on machine learning and the associated pipeline (data creation, preparation, etc.)

2 QoE, QoI and QoS overview

2.1 Definitions and their reference to slicing

This section serves as a reference point to fundamental terminology related to quality of experience.

Quality of Experience (QoE) has been defined by [16] as the degree of delight or annoyance of the user of an application or service. End-user aspects of QoS including QoE should be a set of QoS and performance measurements.

Quality of Service is defined by [15] as the totality of characteristics of a telecommunications service that bears on its ability to satisfy stated and implied needs of the user of the service.

Quality of Perception (QoP) or User-perceived QoS and Quality of Experience (QoE) are two main user-centred approaches.

QoE takes into account quality indicators like service or network based reliability, availability, scalability, speed, accuracy and efficiency.

Service availability is the ability of a service to be obtained within specified tolerances and other given operating conditions, when requested by the user. In [15], defined as Service Accessibility, ideally is measured from the knowledge on the individual availability of each network element involved in the session set up or transmission paths.

Service continuity is the ability of a service, once obtained, to continue to be used under given conditions for a requested duration, defined as Service Retain ability in [15].

Service accuracy is the degree to which a service is provided without excessive impairments, once obtained, regarding content integrity and service usage comfort, included in Service Integrity concept as defined in [15].

Service speed is the degree to which a service is provided without excessive impairments, once obtained, regarding speed concerns and is included in Service Integrity concept as defined in [15].

A network performance is the ability of the network to transmit service and contributes to the QoS perceived by the customer.

Service utilization category is the quantification of service usage: number of customers, number of sessions, total duration of sessions, etc.

E2E cognition requires well defined QoE and QoS to be granted.

QoE is therefore influenced by the delivered QoS (measurable with objective metrics) and the psychological factors influencing the perception of the user. The same QoS level might not guarantee the same QoE level for two different users.

QoE is different from the traditional quality of service (QoS) metric which measures the quality of the network layer services (e.g., packet loss, latency, etc.). In QoE, it is the customer experience that matters. The QoE measurement should consider the individual and dynamic end user's opinion. Applications exert different impacts on user QoE. First, from the user perspective, applications are of different importance to different users and also have diversified network-level QoS requirements. Different application-level QoS performances bring different effects on user QoE. Many contributing factors of QoE change

over time so they are highly time-variant. It is inadvisable and far from reality to measure the QoE once forever such as the traditional methods which try to estimate users' QoE based on preliminary theoretical studies. In order to prevent QoE degradation, it is necessary to monitor the status of each network element in the E2E path of a user session. Ideally, each network element needs to be monitored in real time.

QoE can be measured as the timeliness of content delivery with respect to user needs for timeliness in a given application. The network QoS parameters (i. e. bandwidth, delay, packet loss rate, etc.) are very important that regulate the user perceived quality experience.

QoE Influence Factors (IFs) has been defined in [17] as "any characteristic of a user, system, service, application, or context whose actual state or setting may have influence on the Quality of Experience for the user":

- type and characteristics of the application or service,
- context of use,
- the user's expectations with respect to the application or service and their fulfilment,
- the user's cultural background,
- socio-economic issues,
- psychological profiles,
- emotional state of the user,
- and other factors whose number will likely expand with further research.

It has been widely accepted that QoE is a multi-disciplinary metric, affected by a variety of factors from different fields.

QoE modeling aims to model the relationship between different measurable QoE IFs and quantifiable QoE dimensions (or features) for a given service scenario.

QoE assessment is the process of measuring or estimating the QoE for a set of users of an application or a service with a dedicated procedure, and considering the influencing factors (possibly controlled, measured, or simply collected and reported). The output of the process may be a scalar value, multi-dimensional representation of the results, and/or verbal descriptors. All assessments of QoE should be accompanied by the description of the influencing factors.

QoE includes the complete end-to-end system elements (client, terminal, network, services infrastructure, etc.). Procedure of QoE definition should take into account the specificities of architecture, configuration of network and much more as shown in figure 1.

The most used measure for QoE is the mean opinion score (MOS). MOS is expressed as a single number in the range from 1 to 5, where the value of 1 corresponds to the lowest quality experienced by the end-user and 5 is the highest quality experienced.

QoE	Impairment	MOS
Excellent	Imperceptible	5
Good	Perceptible, not annoying	4
Fair	Slightly annoying	3
Poor	Annoying	2
Bad	Very annoying	1

Estimation of QoE through a prediction model has been an area of significant interest of researchers. Within the artificial intelligence (AI) area, machine learning techniques have been the prime focus for developing objective QoE prediction models. Therefore, several machine learning models have been investigated, such as Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), Decision Tree (DT), Random Forest (RF), Neural Networks (NN), and several more variations.

QoI of each data source mainly depends on three factors:

1. errors in measurements or precision of the data collection devices,
2. noise in the environment and quality of data communication and processing (including network-dependent quality of service parameters),
3. granularity of the observations and measurements in both spatial and temporal dimensions.

The attributes of quality of information are following:

1. Timeliness means the speed at which the information is received. Timeliness in receiving is a determinant of better quality
2. Appropriateness is the suitability matching of the receiver and the information, more the suitability of the information to the receiver, better its quality.
3. Reliability of information is a key attribute of quality.
4. Accuracy - is the correctness of the information. Normally, the higher the accuracy of the information, the better is its quality.
5. Completeness is the measure of comprehensiveness. It is required to ensure that the information provided gives the complete picture of reality and not a part of it.

Quality of information (QoI) should be considered in case of machine to machine communication/interaction.

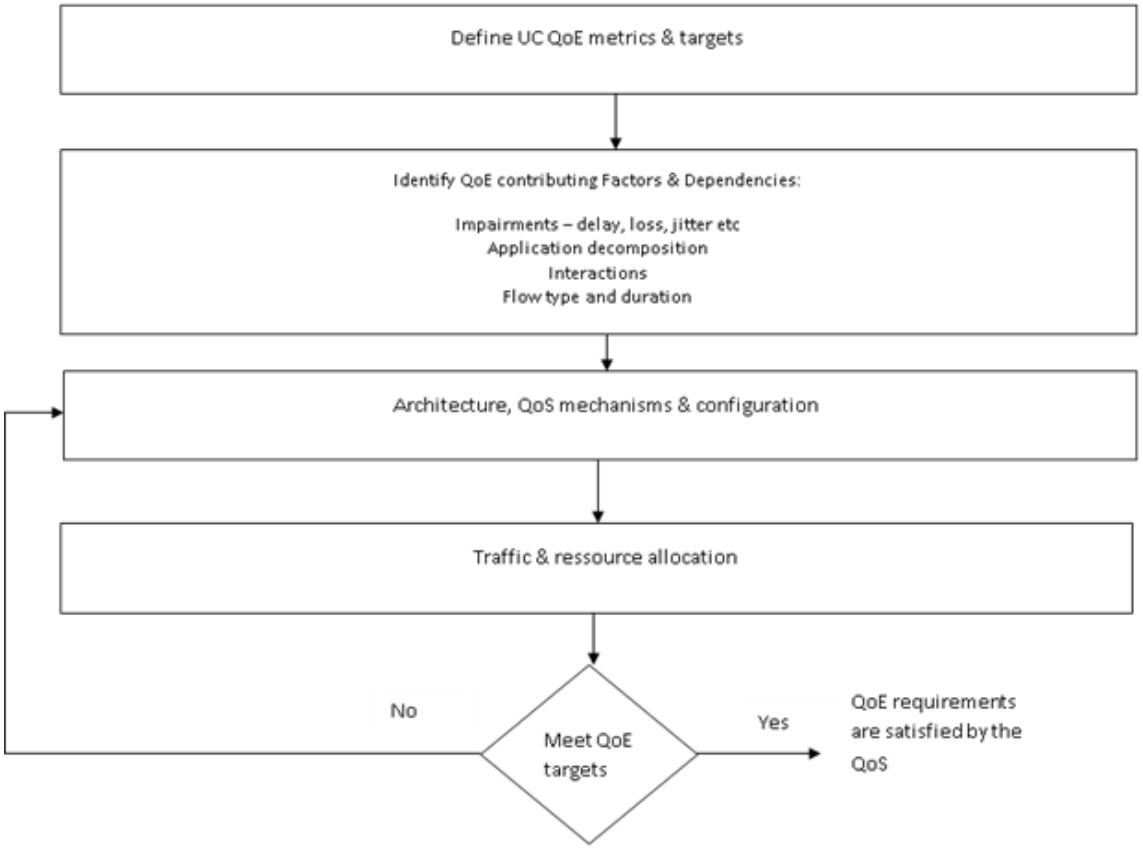

Figure 1 Procedure of QoE definition

Customer satisfaction is largely driven by QoE and not QoS. The key point is that measures of QoE should concern user performance based on actual usage of a service. There will be different user performance measures for different network services because of the different reasons and situations for use. There are many factors, which give variations to the end-users experience, e.g. network type, number of concurrent users in the same location at the same time.

The end-to-end performance that finally drives the user experience can be best reflected by evaluating the QoE. While QoS is based on technical and lower layer parameters associated with physical network elements (e.g. multiplexing, signalling,...) and physical media (e.g. wireless, copper pairs, fiber optics,...), QoE more often uses events on application layer. QoE goes beyond the pure technical measures and considers the end-customer, who interacts with the device, its user-interface, the network, and the service behind it. QoE is providing an end-to-end view and is often based on perceptual feedback given by a user.

In principle, QoE is measured subjectively by the end-user and may differ from one user to the other. The same QoS level might not guarantee the same QoE level for two different users.

A new slice to currently established set of slice instances could cause some problems with satisfying QoS/QoE and isolation level in all slices. Even if resources are available, slices can affect each other. In the RAN, one can see this problem by interleaving communication channels in the frequency domain, which degenerates SNR (signal-to-noise ratio) and consequently BER (Bit Error Rate), throughput as well as causes packet loss, jitter, etc. The objective approach uses some algorithm or mathematical model(s) to predict the QoE by

collecting relevant information from the various layers. The objective techniques that are used should faithfully reproduce the individual QoS factors to the overall QoE score.

Objective QoE models

- media-layer estimates the audio/video QoE by using the actual media signals as their input
- packet-layer utilizes only the packet header information for QoE estimation, which describes them as in-service nonintrusive quality monitoring approaches.
- bitstream-layer utilizes bitstream information as well as packet header information
- hybrid are conceived as a combination of the previously described three models.
- planning models do not acquire the input information from an existing service, but estimate it based on service information available during the planning phase.

Different applications have different QoE requirements (also including different QoS-dependencies), necessitating different QoE models, monitoring and eventually, different QoE management approaches. As shown in figure 2 relationship between QoI/QoS/QoE agreed in SLA contract can be treated as input parameters allowing comparison with QoI/QoS/QoE coming from monitoring.

Depending on different services in SliceNet use-cases, the QoE/QoS/QoI in these use-cases focus on either human-to-machine communications in eHealth or machine-to-machine communications in Smart Grid and Smart City. From the QoE's perspective, the use-cases are concerned with the application-level quality in relation to the Service Level Agreements (SLAs).

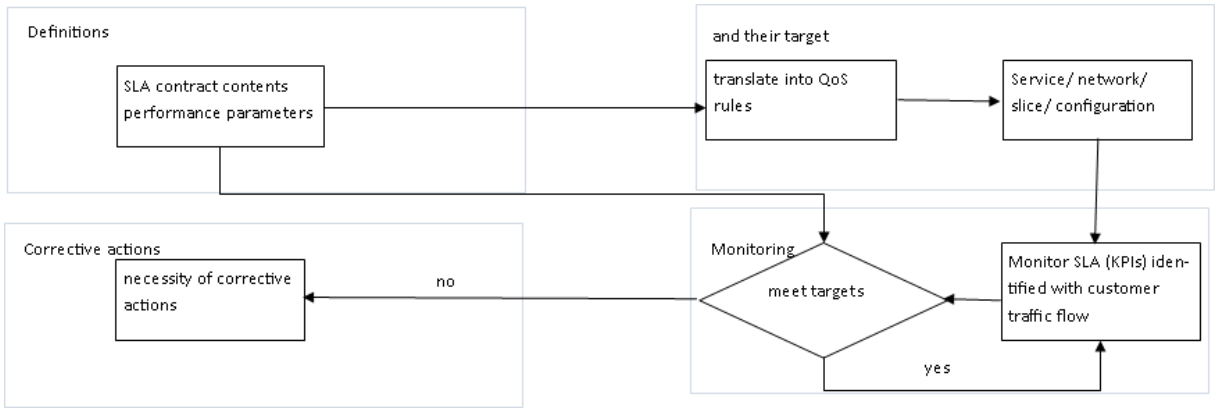

Figure 2 SLA QoI/QoS/QoE management framework

Traditionally, network or services availability is perceived as one of the dimensions of QoS. So far, reliability-related metrics are agreed in SLAs under a general QoS umbrella. It is most commonly expressed as the 99.999% availability requirement, sometimes accompanied with a mean recovery time. The availability agreed in an SLA can be calculated as a mean value for a long time period. Such an approach may be insufficient for some applications and customers. Availability measured in a long time period does not distinguish between frequent short-lasting failures and rare but long-lasting outages. Customers may be interested in the maximum downtime and mean time between failures to show the impact on QoE.

2.2 QoE monitoring

QoE monitoring should include as an input the QoS measurement that is a collection of QoS metrics for the QoE evaluation function. Monitoring architecture should fulfil the following criteria:

- possibility to obtain real time information about the application that the users are using and the QoS status.
- possibility to maintain a data mining scheme that can predict a user's preference/expectations toward the applications in use.
- should manage the communication resources based on the QoS status and the predicted preference/expectation to maintain a satisfactory QoE.
- the inputs should contain a specific user and a specific service in use, and the output is this user's expectation toward this service.

Continuous monitoring of the QoI/QoS and QoE levels for each service flow is required.

Monitoring should be implemented at different processing points at NSP level related to different use-cases in the network collecting measurements related to the behaviour of the network or the service. These measurements are fed into a QoE model, responsible for modelling the QoE at DSP level.

Identification of service relevant QoE indicators and their affectation by QoS indicators are service and application dependent. General rule does not exist for QoI/QoS/QoE mapping. Deep analysis of specific KPIs impact on the user perception is mandatory.

QoI/QoS/QoE mapping is dependent on use-case so it is rather better to provide open framework able to integrate different ways of implementing the required mapping.

It could be a threshold function comparing results with thresholds and if results not exceed the thresholds then quality parameters are acceptable but different KPIs have different impact on QoE so weights are necessary. Information related to KPI calculation can come also from external data i. e. weather conditions, traffic jams etc

A structured approach to QoE can be achieved by applying the clause, as proposed in [15]:

IF <service situation>;

USING <service prescription>;

WITH <technical parameters>;

THEN <user experience>.

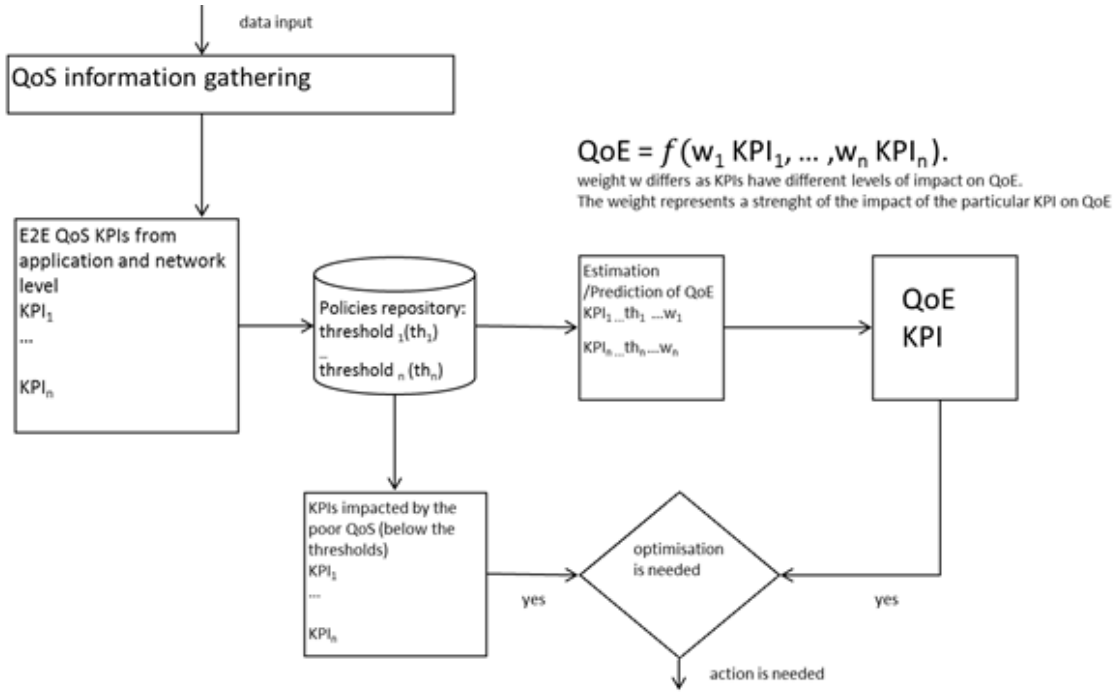

Figure 3 QoS/QoE framework

The implementation of network slicing will determine how reliable the performance guarantees will be. When performance falls below desired levels, it will be a transient event of some short duration: A period of low QoS surrounded by desired QoS. This means that the QoE will follow the same pattern: applications will temporarily lose the required network support and experience will be degraded for a short time.

The QoE metrics will need to capture this transient aspect of degradation. Averages over a minute will not capture it, we must add a summary with the minimum QoE over the minute.

QoE should be calculated periodically, rapidly and very often to be able to capture even short duration of poor quality or performance.

According to [20], key parameters impacting the user are delay, delay variation and information loss. Information loss is not limited to the effects of bit errors or packet loss during transmission, but also includes the effects of degradation introduced by media coding for more efficient transmission.

The delay and packet loss rate are two main QoS parameters that required to be handled in an efficient way in order to improve the user perceived QoE. The delay is an essential parameter that can be caused dropping of packets. Generally, when a packet is arrived after the end of threshold timer, then a packet does not consider in real time application (e.g. audio /video), and it is believed to be lost. For eHealth use-case, QoE is impacted also by problems affecting audio and video quality, which can be categorized as follows:

1. Video signal impairments including compression artefacts and noise;
2. Audio signal impairments including compression artefacts and noise;
3. IP transmission impairments including packet loss, packet delay (or latency), jitter (or packet delay variation), and out-of-sequence or duplicated packets.

Audio quality assessment may be affected by background noise and type of equipment. For video, transmission parameters such as loss or delay will result in video artefacts impacting the media quality, which may in turn be translated to end-user QoE.

The following metrics with associated thresholds are the examples to take into account in QoE calculation. The following thresholds are proposed based on Orange experience.

Attach success rate should be not lower than 98%.

Percentage of session time with audio quality < X (eg. 3 on MOS scale)

Percentage of session time with video quality < X (eg. 3 on MOS scale)

Percentage of session time with combined audio/video quality < X (eg. 3 on MOS scale)

Percentage of sessions experiencing one or more Degraded Service Quality Events event < X%

The end-to-end latency is an important performance parameter for operating 5G network. In some scenarios, for example for Smart Grid, if end-to-end latency is insufficient, the 5G network customer cannot obtain guaranteed network performance provided by the network operator. If high end-to-end latency are measured, it is also of benefit to pinpoint where in the chain from application to UE that the latency occurs.

One of the important metric in slice environment should be how many percent of UEs' Service Experience (i.e. UE QoE or UE Service MOS) is satisfied in a slice?. Satisfaction for this metric should be not lower than 98%.

Service Success Ratio [%]=(number of successful activities)/(number of attempts) × 100 should also take part of QoE calculation

Generally speaking short sessions testify a low quality of experience:

Service activity duration [s] $=t_{end} - t_{start}$

Session Drop has a direct impact on customer dissatisfaction and this metric should not be bigger than 2%.

Handover success rate should be not lower than 98% and Interruption time not bigger than 15ms in 5G Standalone Architecture (SA).

Packet loss is typically caused by a poor quality network, such as high bit errors rates (BERs) on links or network congestion. Packet Loss can also be caused during handover from one cell to another, sometimes resulting in disruption of service. In this regard it is necessary to monitor the packet loss caused due to the handovers and ensure that the packet loss does not violate certain thresholds.

The higher the video transmission success rate and the video quality MOS value the better is the experience of the user.

In addition to the metrics specific to particular services, there is a need to monitor more specific KPIs for a virtualized network environment i. e. for any running VNF instance: collect information regarding the usage of computation, storage, networking resources; for VM activation reliability: symptom of infrastructure issues under production load; VM's overload rate and VM's failure rate; NFV components: failure rate, detection time, restoration time, success rates of the detection and of the restoration, impact per failure.

For QoE supervision (see Figure 4), we need to know user (UE) attached to different slices and network segment ("ObservationPoints" that will be described in section 5) from which we will collect/extract QoS metrics.

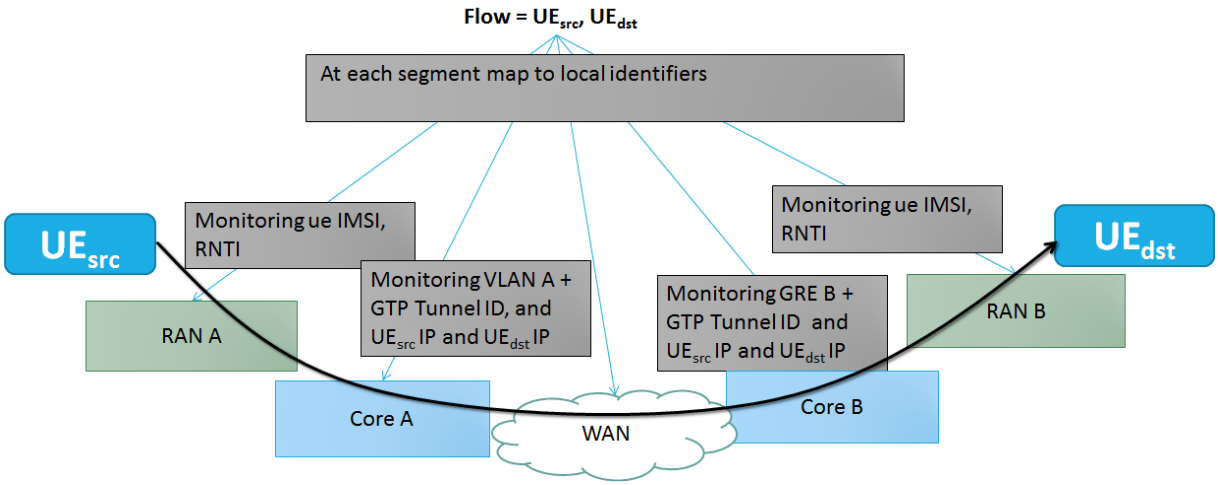

Figure 4 E2E flow monitoring

2.3 Sensor definition

A sensor is an entity that somehow senses and reports one or multiple variables in a system by accessing the data of interest through some API. A QoE Sensor is a "logical/virtual" sensor.

A sensor is a virtual or physical entity such as a device, a program, a module, or a (sub)system that senses (i.e., captures) and periodically reports one or more system values. In the context of 5G slicing, in particular, special QoS and QoE sensors reside at key points in the network for the purposes of collecting data about network activity, and finally processing and reporting system quantitative metrics and user experience qualitative values, respectively. The output of QoS and QoE sensors can be used as input to other sensors (specifically, QoS-to-QoE and QoE-to-QoE sensor input) as well as to separate entities (actuator) that may proceed into any possibly required actions in response to the input from sensors. Examples of QoS sensing reports include raw wireless base station load metrics (e.g., the number of connected UEs or traffic load) or processed metrics like the average signal quality of the connected UEs in a base station or a group of base stations. QoE sensors, on the other, can be seen as "more enhanced" than their QoS counterparts in the sense that they report only processed result values about the perceived quality by users (UEs, or even DSPs and Verticals as a whole). This means that QoE sensors may be not just stand-alone entities, but -most likely- a series of collaborating applications (hence, virtual/logical entities) that take input from QoS and possibly even other QoE sensors lying at any level, even at the highest levels such as patients' critical condition metrics or doctors' preferences for HD video in the context of an eHealth use case scenario.

Depending on the level of the sensing, QoS and QoE sensors reside either at the physical infrastructure level where they are owned and controlled by Network Service Providers (NSPs), or at the level of Digital Service Providers (DSPs), or at the top level of the Verticals. Note that the only physical QoS sensors that can exist are the ones owned and controlled by

NSPs. The rest of the QoS and QoE sensors are naturally logical/virtual entities placed for sensing and reporting at one level below (if allowed by SLA terms) or at the same level. Also, note that only the input from lower or same-level sensors can be leveraged for corresponding actions, i.e. DSP-level actions can leverage NSP and DSP sensors, and vertical sensors can leverage NSP, DSP, and their own level sensors.

Sensors access monitoring data of interest through some Monitoring Framework (MF) API and (optionally) process them. Such raw and processed/augmented/structured data QoS like statistics or QoE values can become available to other sensors and actuators via the MF API that, hence, serves also as a data filtering, store, aggregation and communication pipeline (see section 4.2). In this sense, we envision QoS and, particularly, QoE sensors as special applications on top of an MF API, as portrayed in Fig. 22 later on in Section 4.2. For more on the relation of between sensors and an MF, the reader may refer to Section 4.3 “Monitoring in Support of Cognitive Slice Management”.

3 QoE, QoI and QoS for SliceNet use cases

The main objective of this section is to put in the context of SliceNet use-cases and their requirements with respect to QoI/QoE.

3.1 Smart Grid

This section provides a short overview of the Smart Grid UC (section 3.1.1) and what is the key technical issue to be solved by the service provider (section 3.1.2). Thereafter, it describes the QoI/QoE challenge (section 3.1.3), the involved QoS metrics (section 3.1.4) and the required sensors to deliver the QoI/QoE and QoS information (section 3.1.5).

3.1.1 Use case overview

The Smart Grid use case ultimate goal is to validate an advanced self-healing solution for electric power grids. Shortly, from the power grid perspective, two major steps are involved - the first one is on detecting and isolating the electric failure, whereas the second one is to efficiently reconfigure the power grid and therefore minimize the service downtime.

Intelligent Electronic Devices (IEDs) are used to sense the electric power grid and guarantee that it is configured accordingly and rapidly. To achieve this, the IEDs require a reliable communications network service to allow ultra-low latency communications between them. A specific network communication protocol - IEC 61850 GOOSE - is used to establish the IEDs communications in order to protect and reconfigure the power grid. This is the most challenging requirement posed by the scenario on the service provider network - a reliable service with ultra-low latency capabilities.

Additionally, besides the intra-IEDs communications for the power grid protection and reconfiguration, this scenario also requires a communication service between the IEDs management system and the IEDs for management purposes, such as configuration, remote diagnosis, etc.

From a business point of view, the Vertical will subscribe the end-to-end services (Ultra Low Latency Service and Management Service) from a DSP, indicating the services endpoints (IEDs and Smart Grid Management Server), as well as the associated requirements. Associated with each service subscription is a Service Level Agreement (SLA) which indicates the delivery obligations of the DSP, as well as the counter-measures if there is a violation. It's fully abstracted from the Vertical how exactly the DSP delivers the contracted services, as well as the involved NSPs. Further information about the SliceNet business roles is defined in [18].

Figure 5 provides an overall perspective of the Smart Grid UC, including the Vertical devices (in blue), the DSP actor (in red) and the subscribed end-to-end services - Ultra-Low Latency Communication Service (dashed line) and the Management Communication Service (continuous line).

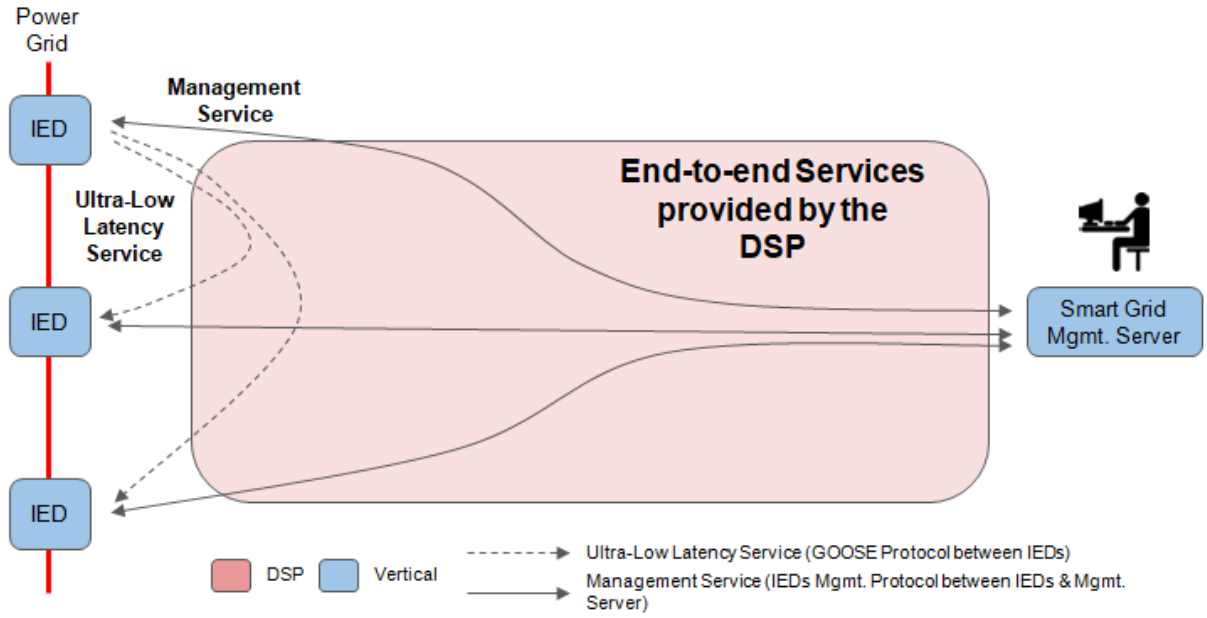

Figure 5 Smart Grid Use-Case High Level Overview

Further details about the Smart Grid UC can be found in [19].

3.1.2 Problem definition

As described in the previous section, the problem space addressed in this scenario is reliability related, especially when referring to the Ultra-Low Latency Service to deliver the communications between the IEDs. Reliability is generally considered as one of the most important quality attributes in Smart Grids.

As described in the previous section, two different services are required by the vertical to address the self-healing scenario in the power grid:

1. Ultra-Low Latency (ULL) Communication Service - delivers the GOOSE protocol messages between the smart grid IEDs with very low latency (maximum 10 ms);
2. Management Communication Service - delivers the management messages between the IEDs management server and the IEDs in the field.

Although the above mentioned communication services are used in the same use-case, they bring very different network requirements and human-machine interactions. For example, the ULL Service requires low-latency in the network, whereas the Management Service does not need such type of challenging feature. Regarding the human-machine interactions, the ULL Service is totally machine-machine (to transport the GOOSE protocol between IEDs), without any human involved in the process, whereas the Management Communication Service requires humans to interact with it (to configure the IEDs, retrieve logs from the IEDs, etc.). Therefore, due to the very different natures of these services, the strategy is to provision different network slices for each service. Below are detailed the subscribed vertical services and how these are mapped to network slices.

End-to-End Ultra-Low Latency Slice

For the ULL Communication Service subscribed by the Vertical, an End-to-End Ultra Low Latency (E2E ULL) Slice is used, as illustrated in Figure 6. A complete 5G mobile network is

required to deliver this E2E slice - including RAN and Mobile Core components. The slice endpoints are the SIM cards connected to the IEDs 5G modem.

As discussed in [19], the E2E Slice is a logical concept, managed by the DSPs, which is materialized in Network Slices (NSs) instantiated at each NSP (which is the business entity managing the NS physical/virtual resources). In this particular use-case, due to the IEDs geographic distribution, the E2E ULL Slice is instantiated using RANs from two NSPs (but a common Mobile Core). Shortly, in terms of decomposition, the E2E ULL Slice, illustrated in Figure 6 (grey), is translated in the following NSs:

1. RAN + Mobile Core NS **@NSP1** (green);
2. RAN NS **@NSP2** (yellow).

It's important to mention that the E2E ULL Slice (subscribed by the Vertical) translation to the NSP NSs is transparent to the Vertical. In other words, depending on the Vertical requirements and on the available NSP NS offers, the DSP decides, during the E2E Slice instantiation phase, which NSP NSs will be required. Therefore, from the business perspective, the Vertical only “sees” the SLA with the DSP. On the other hand, the DSP must manage contractual relationships with the NSP (or NSPs) providing the NSs.

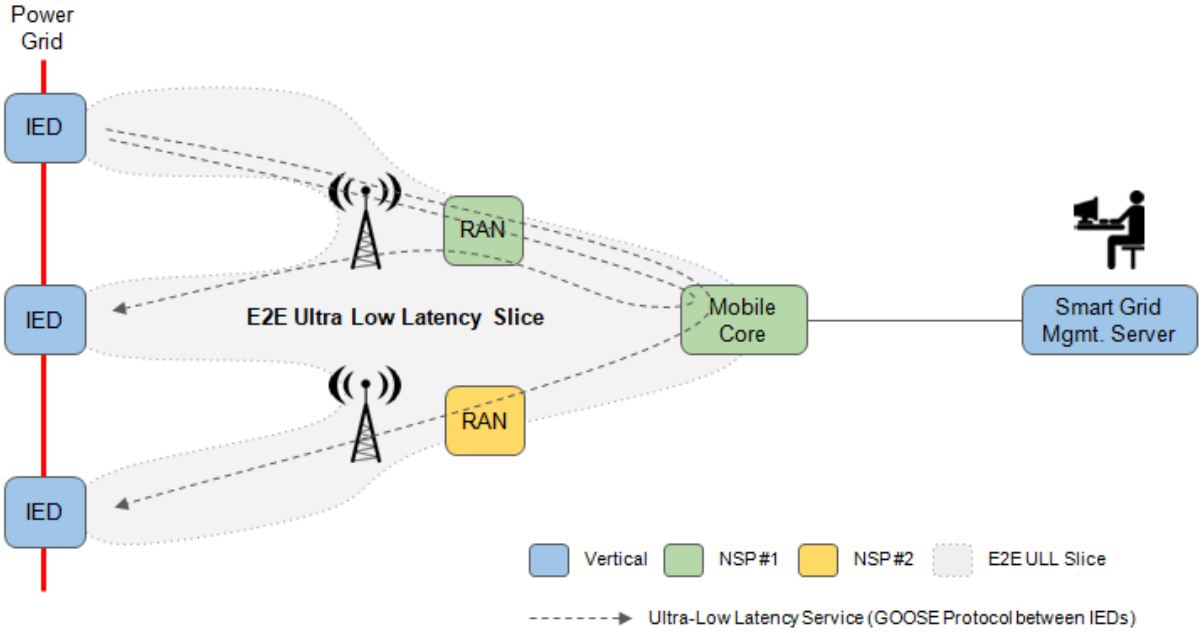

Figure 6 Smart Grid Use-Case - E2E Ultra Low Latency Slice

The DSP must be able to guarantee that the SLA obligations are met towards the Vertical. In this case, for the E2E ULL Slice, it should guarantee that the IED - IED communications over the 5G mobile network delivers **latencies below 10 ms** and that **no packets are lost**. These are the **QoS metrics**/indicators that must be delivered by the DSP towards the Vertical.

End-to-End Management Slice

For the Management Communication Service subscribed by the Vertical, an End-to-End Management Slice is used, as illustrated in Figure 7. In this case, the slice endpoints are, on one side, the SIM cards connected to the IEDs 5G modem, and, on the other side, the Smart Grid Management Server.

The involved actors and NSP NSs are the same ones as in the E2E ULL Slice, with the main difference being that, in this case, one of the slice endpoints is the Smart Grid Management Server.

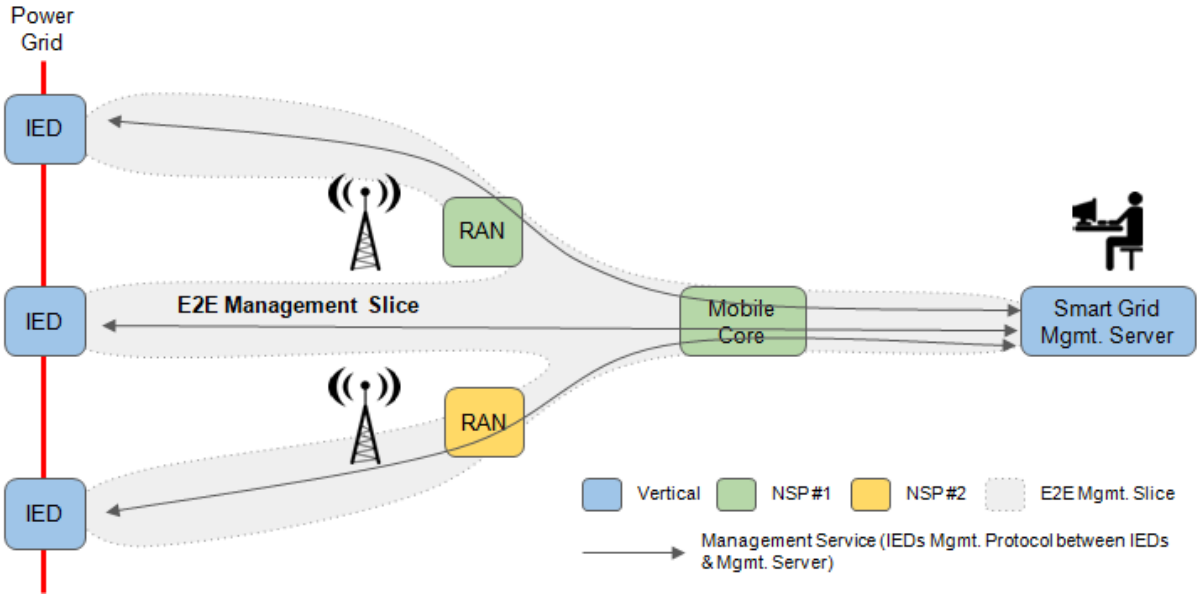

Figure 7 Smart Grid Use-Case - E2E IEDs Management Slice

When compared with the E2E ULL Slice, in this case the QoS requirements are more "relaxed". **Network latencies around 100 ms** should be acceptable and, of course, **without packet losses**.

3.1.3 Use case related sensors for Slice, Resource and application

The Smart Grid requires sensors to retrieve traffic-related information from the several network points and calculate the QoS (traffic-related) metrics described in section 3.1.4. Therefore, network (passive and active) probes are required to provide traffic information at the slice level. To support the foreseen slice management dynamicity, the network probes should be dynamically instantiated, reconfigured, removed, etc., according to the service and slice needs.

At least two options, both supported by the SliceNet system architecture, are available for the network probes deployment:

1. **Conservative approach** - in this case, the NSPs that provide the NSs for the DSP to compose the E2E Slice are not offering the DSP with the capability to dynamically deploy software-based network functions at the NSP NS. In this case, it should be the **NSP managing the virtual traffic probes** deployment and configuration, collecting the traffic information and processing the information to compute the NS-level QoS metrics. Thereafter, the computed QoS metrics are exposed to the DSP and the latter will compute the E2E Slice QoS metrics accordingly.
2. **Progressive approach** - in this scenario, the NSPs that provide the NSs towards the DSP are also exposing the capability for dynamic deployment of software-based network functions (e.g. virtual probes). Therefore, it is under the responsibility of the

DSP to manage the lifecycle of the virtual traffic probes in the several NSPs that compose the E2E Slice.

The developed network probes are able to collect information from ongoing Smart Grid traffic sessions without injecting new data packets on the network - **passive network probing**. Furthermore, **active probing** should also be provided, allowing the injection of new traffic sessions to collect specific information from the network.

Each E2E slice of the Smart Grid use-case - ULL Slice and Management Slice - requires network probes to collect and process information for each one of the slices. Figure 8 illustrates the positioning of the network probes in the E2E ULL Slice (assuming the virtual probes are deployed and managed by the DSP). In this case, since the packets (GOOSE protocol) are flowing between the Smart Grid IEDs, the probes should be deployed very close to the IEDs.

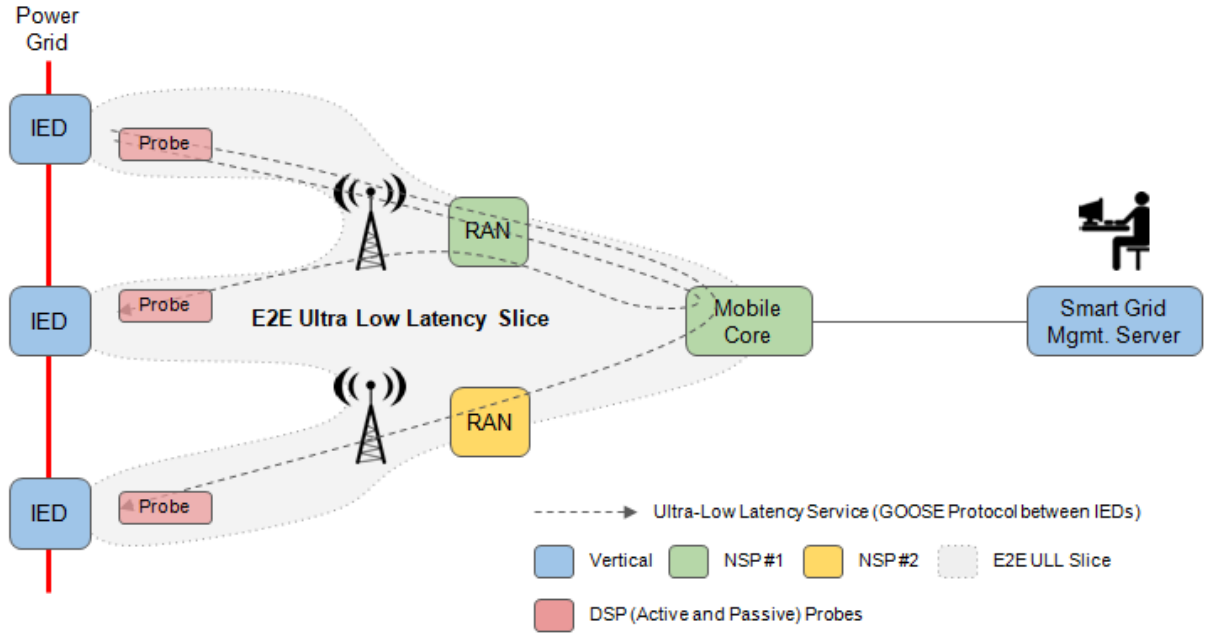

Figure 8 Smart Grid Use-Case - E2E Ultra Low Latency Slice Probes

For the E2E Management Slice, probes should be deployed close to each IED and close to the Smart Grid Management Server that is managing the IEDs (illustrated in Figure 9).

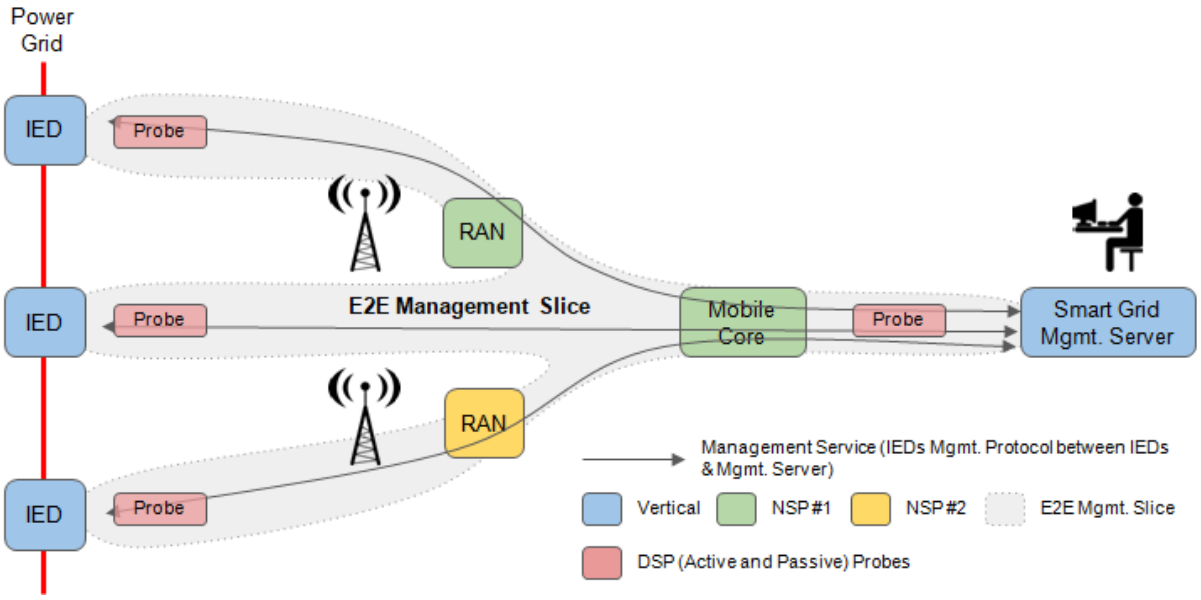

Figure 9 Smart Grid Use-Case - E2E IEDs Management Slice Probes

3.1.4 QoS metrics

Finally, in order to compute the described QoI and QoE metrics, two critical QoS metrics are required for the Smart Grid UC:

1. End-to-end latency: based on the traffic information counters provided by the network probes, the computation of the end-to-end latency is critical to guarantee that the IEDs-IEDs (E2E ULL Slice) and the IEDs-Management Server (E2E Management Slice) communications are not affected. For example, in the E2E ULL Slice, any communication latency above 10 ms (in the GOOSE protocol messages between the IEDs) will compromise the power grid protection and automatic reconfiguration procedures. On the E2E Management Slice, although not so demanding as in the E2E ULL Slice, latency issues will affect the Smart Grid operator experience.
2. End-to-end packet loss: guaranteeing that the end-to-end packet loss is kept within acceptable rates is also mandatory. In the E2E ULL Slice it's crucial to avoid packet losses in the IEDs-IEDs GOOSE protocol communication - any packet loss will compromise the effectiveness of the system to self-heal the power grid. Also in the E2E Management Slice, losing packets will affect the user experience during the IED lifecycle management procedures (e.g. configuration, diagnosis, etc.).

As described in section 3.1.3, depending on which business entity is managing the network probes lifecycle, the collected data processing in order to compute the above mentioned QoS metrics can be done at the NSP or at the DSP level.

3.1.5 QoE/QoI metrics

Typically, reliability is defined as a fraction of unsuccessfully delivered data to the whole volume of the transferred data (i.e. probability of data loss). Moreover, it can be specified in terms of other Quality of Service (QoS) parameters. These include latency, payload size, bandwidth or frequency of data transfers. Any violation of the specified QoS can be perceived as a failure in the Smart Grid system. One of the KPIs used for reliability can be a

Loss of Power Supply Probability (LPSP). If LPSP is low then it results in high reliability of the system.

Average Interruption Duration is computed by dividing the sum of long duration interruptions (i.e., longer than 3 minutes) by the total number of customers. It is a measure of the average amount of time when customers encounter interruptions. Useful information can be timeseries of failure occurrences or mean-time-between-failures (MTBF).

Smart Grid also relies on various software systems, the potential failures in their software components might also have a significant impact on the Smart Grid operations. As an example, it might be a failure during the reading of the smart meter, resulting in obtaining incorrect consumption data (QoE impact).

For prediction of system failures, the probabilistic indicators, such as probability of loss of load are recommended, which represents the probability that there will not be enough power load to satisfy the demand, i.e. the outage occurs.

End-to-End Ultra-Low Latency Slice

Since the E2E ULL Slice does not involve human interactions, but only machine-to-machine communications, the **Vertical feedback** will be critical to understand if the Smart Grid system experience is good or not, as well as to calculate the **Quality of Information (QoI)** metric. Moreover, the **QoI metric** calculation depends on the **QoS information** collected by the DSP and/or NSP, as well as on the experience **feedback information** provided by the Vertical Smart Grid system (IEDs & Smart Grid Management Server). The calculated QoI metric will allow the DSP to create a historical view about the set of QoS metrics that should be delivered for the Smart Grid specific requirements. This information can be used either in run-time to adapt the delivered slice QoS metrics or in design time to create more robust slice offers.

If the calculated QoI metric is not acceptable, there might be several causes associated with this behaviour:

1. If the QoS metrics are within the agreed SLA boundaries, it means that the Smart Grid devices might have a malfunction; another alternative, still assuming that the QoS is within acceptable ranges, is that the requested QoS parameters from the Smart Grid operator are not “enough” to satisfy the use-case requirements;
2. If the reason for the unacceptable QoI metric is a QoS related violation, then the DSP must check which NSP is not delivering as expected.

In this particular E2E Slice, since there is no human intervention, calculating the Quality of Experience (QoE) is not relevant.

End-to-End Management Slice

In this case, the Smart Grid Management Server operator is responsible for interacting with the IEDs for management purposes (e.g. remote configuration, diagnosis, logs retrieval, … ,). Therefore, the Smart Grid operator feedback is required to estimate the Quality of Experience (QoE) metric delivered on this slice. As in the E2E ULL Slice, the computed QoE metric will depend on the user feedback, as well as on the slice QoS metrics.

3.2 Smart-City

3.2.1 Use case overview

As described in [19], a Smart City consist in metering solution (gas, energy, water), remote monitoring of city infrastructure (pollution, temperature, humidity, noise), real-time traffic information and control, city or building lights management and public safety alerts for improved emergency response times, besides aggregation of these services with very different characteristics. The scope of the use case is to build a live testing infrastructure, deployed in a city street that will aggregate the projects developments and benefits and will increase the streets lighting efficiency, including cost efficiency and performance.

There is a clear separation between the roles and responsibilities, as described in [18]. Orange is playing the role of the DSP and NSP, into a single domain implementation.

The roles and responsibilities into a combined scenario, as it is proposed, it is reflected by the figure 10.

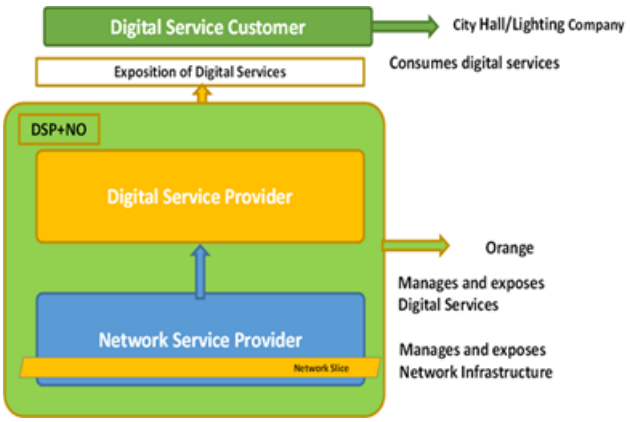

Figure 10 NSP and DSP integration in Smart City Use Case

The City Hall is the entity that is consuming the digital exposed services and collects information about the service behaviour and the DSP + NSP (Orange in this use case) handles the management of the services to the customer and the resources allocation with the network infrastructure, also under the management and responsibilities of Orange in this particular scenario implementation. The testbed is built with respect of this aspect and it will be reflected into the architecture.

The Smart City use case is built based on the integration of several specific components that will provide the use case functionality, as described mainly in [19], the use case approach is based on the following high level figure:

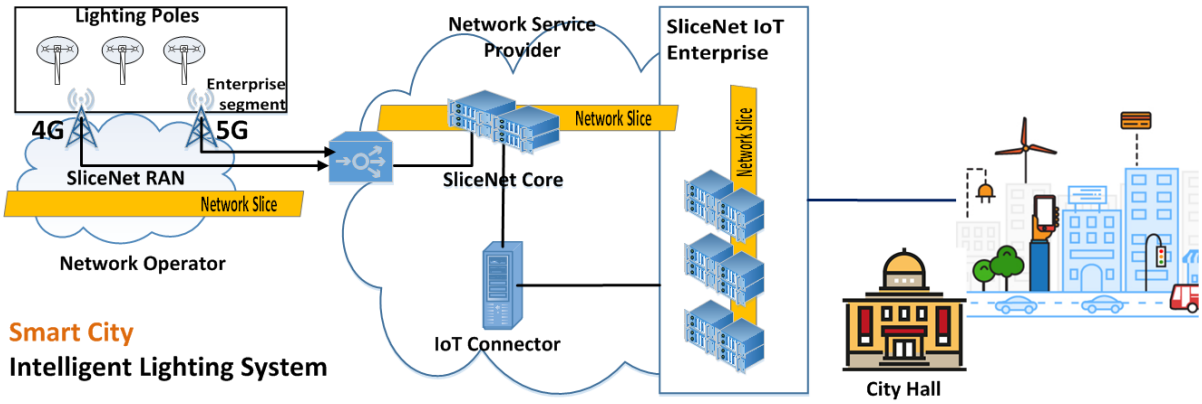

Figure 11 End-to-End Architecture of ORO Smart Lighting System

The Smart Lighting use case consists of several activities and actions related to the vertical, as system control, system software upgrade, new poles integrated in the system, service design.

1. Smart Lighting system control, the responsibility entity controls remotely in real time the lighting pole for the target network(lighting poles device and application)
2. Smart Lighting System software upgrade of lighting pole sensors(a managed upgrade for the lighting sensors, upgrade required by the lighting operator)
3. New lighting poles and devices needed to be integrated in the system(new lighting poles are physically installed in the city
4. Service design by the use case consumer(the vertical, lighting service consumer(DSC) starts designing the service and the service needs)

The implementation of the use case is done with respect of the use case requirements in term of technological needs, the specificity of the use case, as approaches as QoE/QoS, actuation and P&P capabilities, and performance, as described into the table 1.

Table 1 SliceNet Smart City requirements

SliceNet Key Requirements	Smart City
Alignament to 3GPP Ucs	mMTC(mIOT)
QoS/performance requirements	High device density
Multi-domain	No
Mobility/handover Control	No
MEC	No
Enterprise	Yes(extended)
Scalable Slice(resource control/management)	Yes(horizontal/vertical)
Communication pattern/service	unicast/anycast
FCAPS requirements	Yes
Cognition requirements	Yes
Sensors, Actuators, Network functions, Apps	Flow;Resources;Topology;Security
P&P	Yes
Multi-tenancy	Yes

SliceNet Smart City requirements

For sake of clarity, the smart city use case is considered single domain, the DSP and NSP role played by Orange, no request of mobility, low bandwidth, accepting delays within the order of 100s of ms, within mMTC 5G vertical's use case family.

3.2.2 Problem definition

The UC problem is defined taking into the consideration that a Smart City involves the connection of the enormous number of devices, simultaneously to the network. The possibility to connect tens of thousands or millions of devices, sensors / km^2 with specific traffic characteristic as bandwidth, payload size, latency and traffic pattern requires a specific QoS that in case of violation the Smart Lighting services perception (QoE) may be received as service failures for end-users (customers, citizens) or the public entities, Cities Halls or Public Administrators.

In general, a Smart Lighting use case has its own application software that relies on the end-to-end network capabilities and performance service (QoS), as for the mMTC vertical's the problem solving is to be able connect and provide communications services to the devices, as a balance between the offered QoS/QoE and consumed/allocate resources for the service, based on the criticity.

A proper description of the Use Case requirements in terms of QoS/KPIs, communication needs and traffic patterns, together with the network slice resources allocation, with different levels of continuously monitoring and data collection and processing capabilities (ML) allows the measurement of the end-to-end system performance (correlated with the customer satisfaction).

3.2.3 Use case related sensors for Slice, Resource and Application

The Smart Lighting use case relation with slice, resource and application is identified by the brief description of the use case network elements (NFV/VNFs), components and specific application, including an infrastructure description on which the use case can work on:

1. Enterprise Segment[DSP] virtualized components
    - Lighting Poles including the radio modules (LTE/Cat-M/Nb-IoT)
    - IoT Enterprise applications(Dashboard)
        i. Smart Lighting application(including network, protocols and messages for IoT), tenant based
2. Network Operator virtualized components[NSP]
    - 4G RAN Components(OAI-RAN), with LTE/Cat-M/Nb-IoT capabilities)
    - 4G EPC Components(OAI-Core and HSS)
3. PNF/VNF
    - IoT Connector, IoT Gateway, not relevant for QoS/QoE activities, not part of the slice, required for end-to-end service functioning, viewed as part of the Enterprise segment.
    - network elements for IP transport(switches and routers) and Security functions for infrastructure(Firewalls)
    - cluster of servers for compute, control, management and storage
    - Radio RRUs USRPs

For sake of clarity, the use case related sensors for resources slice and application is based on the following sub-planes monitoring approach:

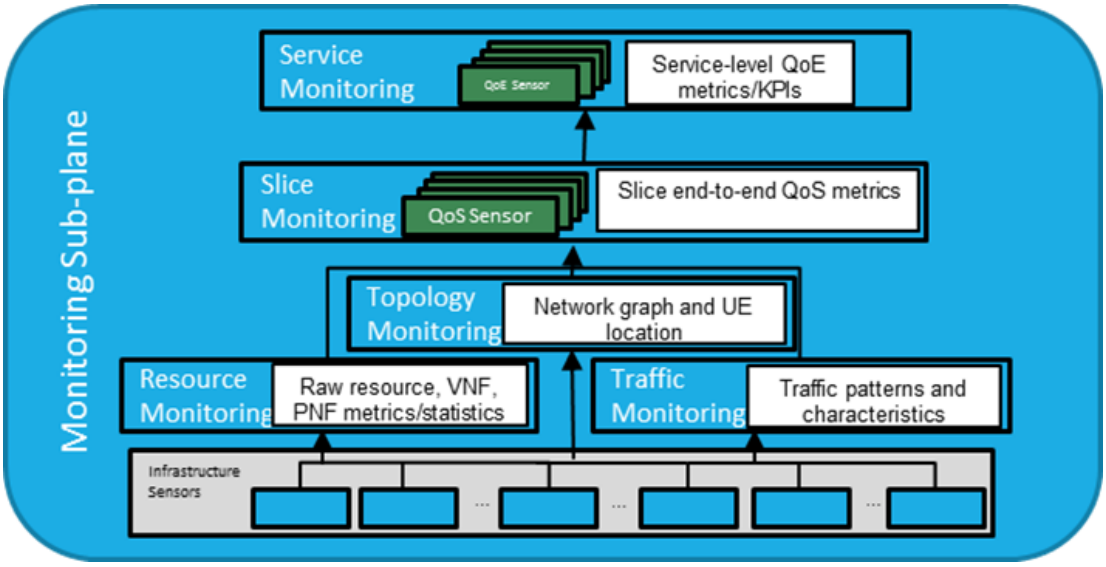

Figure 12 SliceNet monitoring sub-plane

Resource monitoring: set of probes that monitor the entities that constitutes the data plan, the VNFs states that constitute the use case network components. The monitored values are saved into the resource monitoring database.[QoS sensors]

Topology monitoring: collection of the status of current virtual topology, considering that in this use case the UE has no mobility, and in this condition the topology is fixed. [QoS sensors]

Traffic monitoring: traffic flow monitoring, extract and aggregate, including traffic patterns. [QoS sensors].

Slice monitoring: collection of the data related to the slice level performance indicators as bandwidth and latency (between different VNFs, or End-to-End) as a global QoS for the slice. [QoS sensors]

Service monitoring: service instance performance monitoring [QoE sensors]

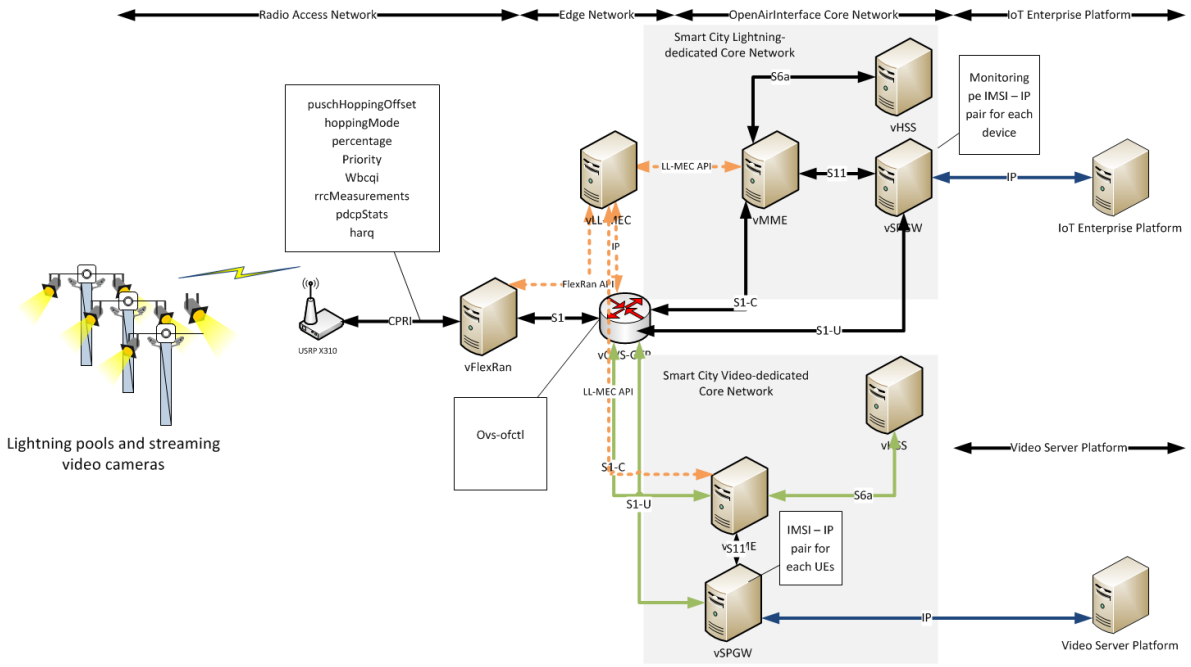

Figure 13 Smart City resource monitoring collection points

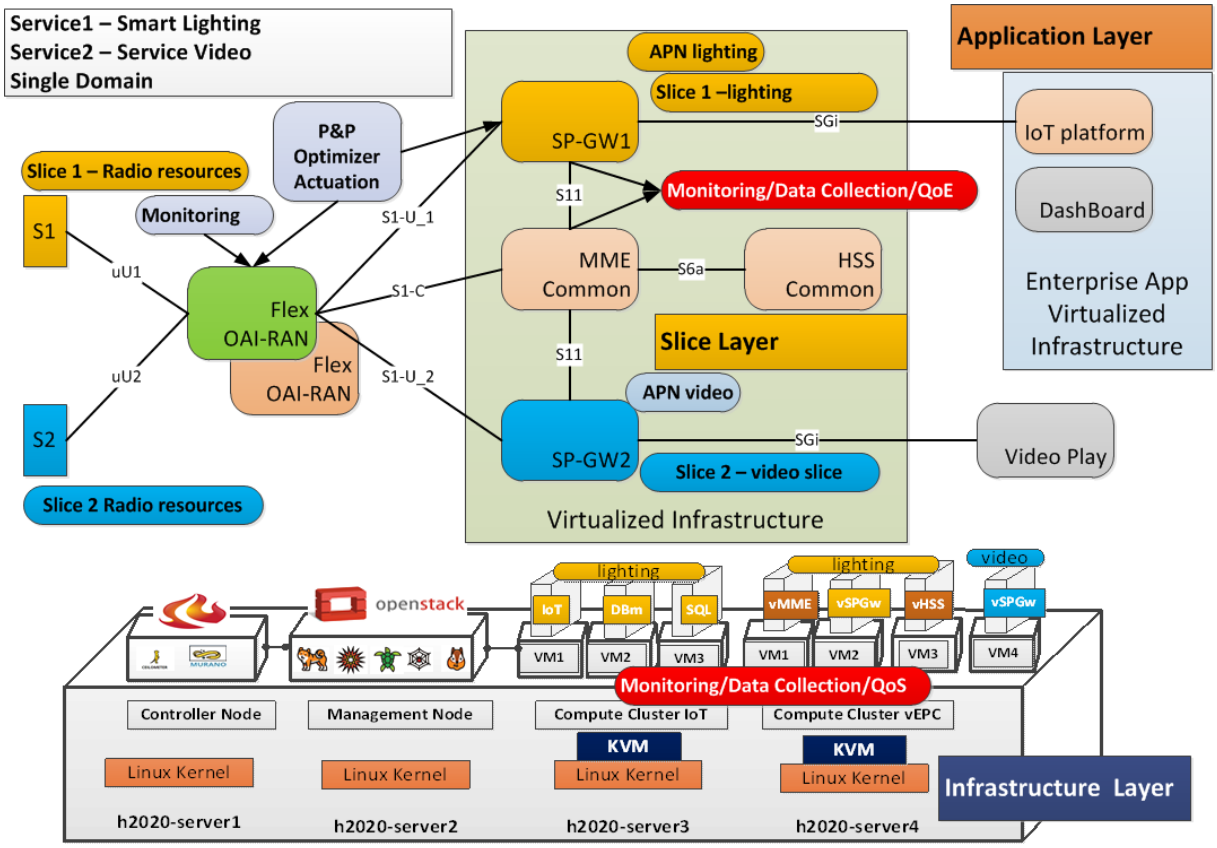

Figure 14 Smart City resource, slice and service monitoring

Resource monitoring:

4. infrastructure sensors(physical metrics): gnocchi-metric, CPU, BW for servers and interfaces, RAM, storage
    - to be seen if it is required to be used as a metric from SliceNet perspective
5. resource monitoring: VNFs, raw data as CPU, RAM, link bandwidth
    - monitor VMs states (vEPC, vRAN) and load, including % of the allocated values

Traffic monitoring

6. traffic monitoring(flows), pattern and characteristics for the smart city use case, by defining two type of traffic patterns:
    - regular traffic: messages(same type of data); size: ≈ 1508B transmitting all data once per lighting poles(the periodicity is once a minute;
        i. Transport protocol UDP; Application Layer Protocol: MQTT; MQTT encapsulated over UDP then in Core over GTP
        ii. command message: up to 70B;(command is ON/OFF/Dimming)
        iii. new on-boarded devices, connecting to the network, that will increase the global traffic per slice, evaluated also in the resource and slice monitoring
        iv. assured capacity of 50kbps/device, that concludes the total slice allocated bandwidth
    - software upgrade of devices/lighting poles; increased traffic per UE, modified traffic pattern up to 300 kbps/UE
7. in order to achieve and demonstrate the mIoT traffic behaviour, from PCs or UEs connected to the network it will be generated a simulated traffic equivalent of 1000 lamps(same total payload values), including on/off/dimming simultaneously
    - due to the lack of 1000's of devices, it cannot be simulated the simultaneously devices connection to the network
8. monitoring SP-Gw through an API(per IMSI) and Flex-RAN through API(per IMSI)

Topology monitoring

9. topology monitoring - it has not been identified a way to discover dynamically the topology for the use case, but the use case is based on:
    - a fix topology(eNodeB; vEPC:vMME, vSP-GW;vHSS; IoT)
    - the UE location is fixed, no mobility required
10. the topology monitoring requires the VNFs status and functioning parameters in range of consumed resources:
    - VMs are up; CPU, RAM, link bandwidth (monitored with gnocchi/ceilometer - to check), exposed through DataBase or API.

Slice monitoring

The Slice monitoring collects counters and eventual alarms from the slice level components, topological known, as total bandwidth consumed per slice from the PGW

Service monitoring

The service monitoring collects alarms from the IoT platform related to the lamp device status: online or offline and test the lamp device end-to-end connectivity from the IoT

platform, the results will be collected into a database, exposed information, correlated with the other monitoring components.

3.2.4 QoS metrics

For Smart City - Smart Lighting use case QoS is analysed on technical measures like service availability, bandwidth, latency and packet loss which can present the negative or positive QoS. Based on that, there are defined values (that are listed in Table 2) in which the use case is working proper and the consumed resources are optimal.

The Smart City QoS metrics are expressed below taking into the consideration the following parameters:

1. service availability, measured as:
2. average bandwidth, measured as:
3. total slice bandwidth: devices number x bandwidth/device (helpful for VNFs system parametrization), measured as:
4. latency: measured as:
5. packet loss: ≤ 0.1%

Table 2 QoS range information for Smart City use case

QoS measured	QoS Range
Service availability	99,99 %
Bandwidth	20-100 kbps
Latency	50-300 ms
Packet loss	<=0.1%

QoS range information for Smart City use case

Optimal values for the QoS measured parameters with respect of the allocation of the resources (physical, virtual) for use case implementation are presented considering only two relevant parameters: latency and Bw.

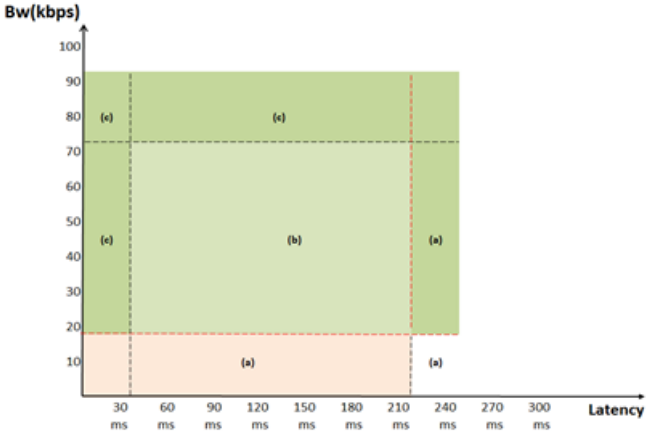

Figure 15 QoS/QoE regions based on latency and BW in Smart City Use Case

The QoS/QoE problem is solved by the existence of the three regions, as:

1. (a) unacceptable region, poor QoS for the customer. The service works with penalty in this area
2. **(b) optimal QoS region, good QoS for the customer, optimal resource allocated for the service, assuring the service QoE**
3. (c) better services QoS than requested by the customer, the customer is happy but the allocated resources are not optimal

Region (b) defines the preferred area where the QoE is assured within feasible resource allocation.

From resource availability perspective, the Smart City use case considers the next metrics to be taken into consideration: CPUs, RAM, virtual links bandwidth, expressed as percentage of usage, as for values between 20% □ 80% of the provision capacity there is no need to scale the machines.

3.2.5 QoE metrics

Regarding QoE metrics for the Smart Lighting use case, this values can be obtained and predict from some of the network QoS KPIs and other external parameters collected from specific APIs and sensors integrated in the architecture of the pillar.

The Smart Lighting QoE metrics are expressed based on a group of parameters that will be collected and weighted according to needs of traffic flows, day and night, on different roads such that can minimize the power consumption and resource utilization while maintaining the city safe from lighting point of view.

In this manner, it will be determined the value of **PDL - Predicted Dimming Level**, a parameter used to set the percent of light produced by the lamps in every area (area – a street or a bunch of streets, depends on its objectives).

The formula that determines value of **PDL** can be considered as below:

PDL = (w1*ADLT +w2 *ADLW)/(w1+w2), where:

- **ADLT** is the coefficient of Average Dimming Level on current Time
- **ADLW** is the coefficient of Average Dimming Level on current Weather
- **Wx** is weight coefficient associated to any above parameter

All the logic and operations to set this rate (PDL) it is held in IoT application system such that can also offer control and visualization to the end user regarding service.

Steps for this process:

1. In the first step, sensors has to collect data for both coefficients from external source (public APIs) and hardware sensors built-in pillar structure.
   - Average dimming level on current time ADLT: using a special API (e.g. timezonedb.com/api) for the local time (sunrise and sunset) correlated with the GPS coordinates registered in the platform for every device/lamp.
   - Average dimming level on current weather ADLW: here it can be used also an API (e.g. openweathermap.org/current) for the city or light sensors on the pillar (to indicate a luminance coefficient).
2. In the second step, collected data is stored in a structured manner to ease the process of correlation from the next step.
3. In this step, collected data will be correlated based on:

- Locations: GPS coordinates
- Lighting zones (table s.3): prioritize if it is a critical/important area (e.g. hospital)
- Time: prioritize (increase dim coefficient if is nighttime)
- Weather condition

Special events in the area (increase priority if it is a crowded area) - detect increasing of traffic and type of the UEs from FlexRAN using LCID (Logical Channel Identifier) that can determine what type of device is connected.

Table 3 Short description of lighting zones

Category	Zone	Description
C1	dark areas	national parks, natural spaces, etc
C2	low brightness areas	rural, small village, dark urban locations
C3	medium brightness areas	urban locations
C4	high brightness areas	town centers

4. In this step, data is analyzed and then applying PDL formula. IoT platform sends the commands to lamps to dim at recommended values.

e.g.:

control_lamp.py - root - <on_message> - [INFO] - Message received: {"method":"setValue","params":"51"}

control_lamp.py - root - <publish_to_gw> - [INFO] - Publish to device

control_lamp.py - root - <publish_to_gw> - [INFO] - Dimming the light at 51%

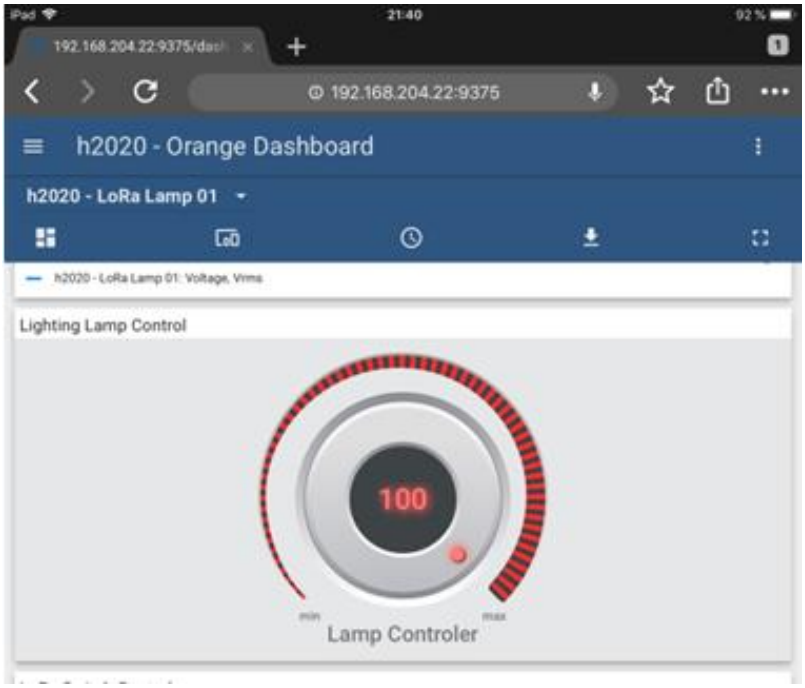

Figure 16 Dimming control panel of IoT platform (tablet printscreen)

It has been identified a list of potential issues (A) that could occur and then affect the OoE/QoI of the use case with a serie of consequences (B), as:

(A)

- Delay in the control packets transmission, as values marked exceeding the QoS limits
- Packet loss of control packets, as values marked as exceeding the QoS limits
- Retransmission (at the transport level) of control packets which impacts on QoI
- Attach request to the network rejected, the devices cannot connect (scalability issue related to density of devices, as the problem definition context) which impacts on QoI.

(B)

- number of LPs that were lighted up, as it is assumed that light up occurs once in a day, impacting the QoE
- number of LPs that were lighted down, as it is assumed that light down occurs once in a day, impacting the QoE
- unnecessary lighted time over per period of time (eg. week or month) additional correlation with Predicted Dimming Level, related to the lighting needs
    - expressed as an wasted energy that is impacting the QoE for city hall perspective
- Total "required" lighting time when the LPs where not lighted over a period of time (eg. Week or month)
    - This represents "insecurity" - QoI
- number of times over a period of time when a technician is usually sent on the field, as through the IoT application is thought that the light was broken, since several repeated packet loss occurred, impacting on QoE, close depend if it is only one lamp in area or more
    - number of times over a period of time when a technician was not sent on the field

3.3 eHealth

3.3.1 Use case overview

Detailed description of EHealth use-case can be found in [19]. In SliceNet project the Connected Ambulance will act as a connection hub for the emergency medical equipment and wearables, enabling storing and real time streaming of video data to the awaiting emergency department team at the destination hospital.

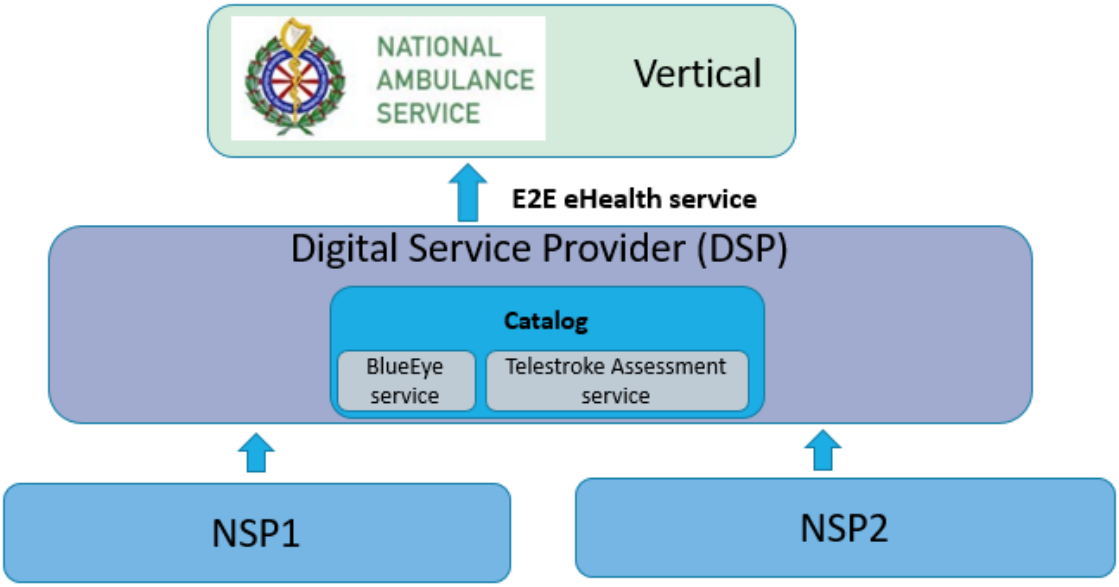

Figure 17 Roles mapping in eHealth UC

BlueEye service requires to monitor the QoS/QoE at the application level during (and after) runtime (E2E latency, rate, loss) and report the experienced parameters to the SliceNet (Vertical Sensor). Within the eHealth use-case, the real-time video streaming service provides a set of requirement ranges for QoS that guarantee an acceptable QoE, from the perspective of the health professionals as defined in [21].

3.3.2 Problem definition

Different network slices have different characteristics and requirements in terms of mobility, latency, and reliability. There are two main procedures in mobility management which are location registration and handover management. Flexible mechanism for selection of mobility management schemes is necessary.

Devices register their locations when they first connect to the network, and then report their location information to the network periodically. To achieve unified multi-RAT access and seamless mobility in 5G networks, multi-RAT coordination is needed for different RATs to share location information of their devices. In legacy network, handovers are mainly event-triggered. The base station controls the user terminals to execute the measurement and report the measured network status information to the serving base station. Flexible handover mechanisms and adaptive handover thresholds should be exploited to support mobility management in service-tailored scenarios. Due to limited network resources and increasingly diversified network services, it is challenging to efficiently provision network resources to network slices with different QoS requirements and also for densely deployed cells, efficient and flexible resource allocation schemes with interference awareness are needed. Key issue is to improve QoE reducing the number of handovers.

During the UE's movement (ambulance movement), received signal strength from all cells is varying. Therefore, if the UE crosses boundary between two cells, handover to the target cell is performed. Handover in mobile network is typically initiated if the signal level observed by the UE from the serving cell (*ss*) drops below the signal level from the target cell (*st*). This condition must hold for the time-to-trigger (TTT) interval in order to avoid ping-pong effect. When a RAN node serving the UE tries to initiate a handover, to select a suitable target RAN

node, besides the radio signal strength, it may also take into account the network overload information and the potential UE waypoint(s) if it has (RAN node ID(s)) of the UE based on the predictable UE mobility information and the current UE location).

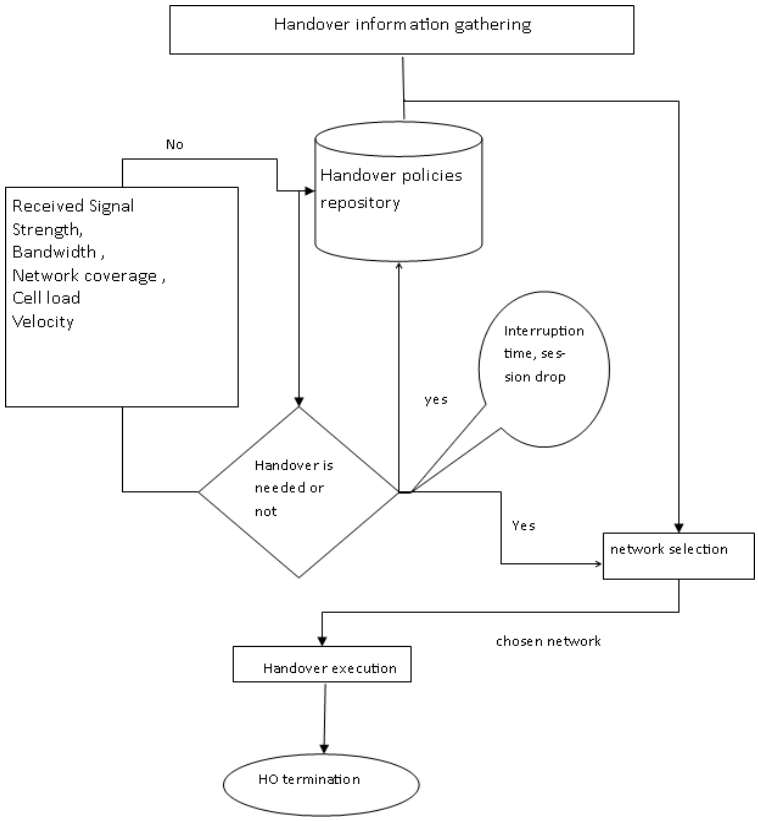

Figure 18 Handover procedure

The challenging requirements in terms of mobility support in 5G are governed by end devices. While the connection characteristics change during movement across multiple cells and points of attachment, the session has to continue. Also seamlessness with respect to the user or application experience may differ, demanding for minimal disruption without loss of information. Load of neighbouring cells and radio technology or specific parameters such as bandwidth and latency or performance requirements of the application in terms of connection reliability and session continuity have to be considered during mobility management.

In order to optimally utilize the handover detach time the user plane switching must start together with the handover command. In other words, when handover notification is received, the new path of the user plane is ready. The prerequisite for this is that the new Tunnel Endpoint ID (TEID) from the new access point is already available.

In order to select a more suitable cell, the prediction on UE mobility will be helpful when the UE mobility is predictable. One of the main goals is to provide fast and seamless handover from one cell (a source cell) to another (a target cell). The service should be maintained during the handover procedure, data transfer should not be delayed or should not be lost, otherwise performance will be dramatically degraded and it will have a direct impact on QoE.

The applicative interruption time is measured at application level between last packet received by the UE on initial layer and first packet received by the UE on target layer.

Handover may impose additional delays and packet losses, resulting in potential loss of session-related content and awkward communication.

The challenge of mobility then is to decide if and when to make this mobility decision and fulfil the goal which is seamless service connectivity. The decision method for mobility should minimize service outages, while at the same time prevent QoE from degrading when users move between cells.

Effective handover management using for example network function mobility analytics, as shown in figure 19, can improve QoE negative impact of cellular network densification due to the imposed excessive handover rate. The final QoE depends on their position in the cell, their mobility, the cell load, interruption time in mobility case and session drop.

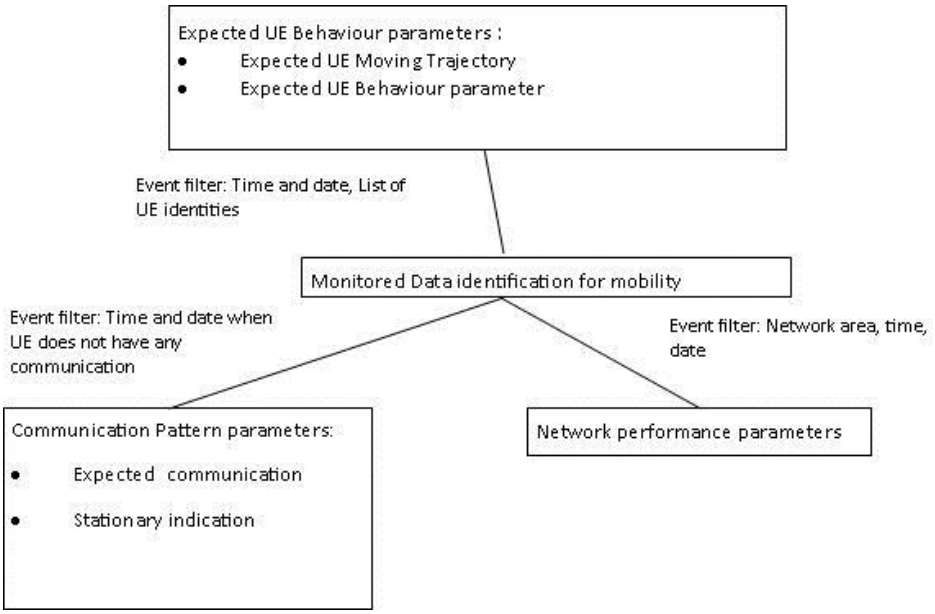

Figure 19 Network function mobility analytics

Network performance parameters should contain at least signal strength, quality of the channel, total load in a base station.

The cell selection during mobility procedure should be based on the QoE.

In 5G [22], Network Data Analytics Function (NWDAF) can be used in Mobility management, Session Management, QoS management and much more. In order to select a more suitable cell, the prediction on UE mobility will be helpful when the UE mobility is predictable. Since NWDAF can learn UE mobility history and discover the laws/patterns of UE mobility, the 5GC can apply the discovered laws/patterns of UE mobility to predict the UE mobility, e.g. the moving trajectory. Therefore, the 5GC can use the NWDAF output to predict the UE mobility and indicate the predicted UE location to the NG-RAN node serving the UE. Then the NG-RAN node can select a more suitable target cell during the handover or redirection.

The Mobility Pattern is a concept that may be used by the AMF (Access and Mobility Management Function) to characterise, and optimise the UE mobility. The AMF determines and updates Mobility Pattern of the UE based on subscription of the UE, statistics of the UE

mobility, network local policy, and the UE assisted information, or any combination of them. The statistics of the UE mobility can be historical or expected UE moving trajectory.

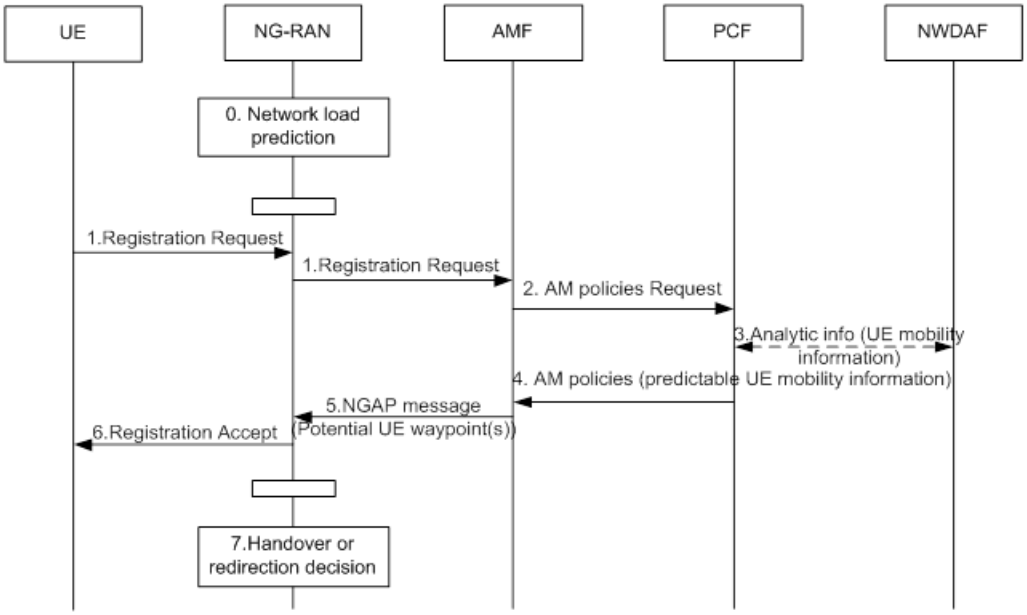

Figure 20 Handover procedure using analytics

1. This solution assumes that network load prediction can be used to assist the source RAN node to select a suitable cell or RAN node.
2. A UE registers to the 5GC.
3. Based on operator's configuration or UE subscription, the AMF may contact the PCF to retrieve Access and Mobility policies including predictable UE mobility information, or contact the NWDAF directly to retrieve the predictable UE mobility information.
4. If the PCF has stored the predictable UE mobility information for this UE, the PCF provides the information to the AMF. Otherwise, the PCF contacts the NWDAF first.
5. The NWDAF performs data analysis on historical UE mobility information obtained from the OAM. The analytical result on UE mobility, e.g. UE mobility pattern, is provided to the PCF. The PCF stores the analytical result in the UDR as the predictable UE mobility information, which allows other NFs to retrieve the UE mobility information directly, i.e. without consulting the NWDAF.
6. The AMF checks the UE mobility information and determines whether current UE mobility can be predicted, if yes, the AMF calculates the potential waypoint(s) (e.g. RAN node ID(s)) of the UE based on the predictable UE mobility information and the current UE location. The AMF provides the potential waypoint(s) of the UE to the NG-RAN node serving the UE.
7. The Registration Accept is forwarded by the NG-RAN from the AMF to the UE.
8. When a NG-RAN node serving the UE tries to initiate a handover, to select a suitable target NG-RAN node, besides the radio signal strength, it may also take into account the network overload information and the potential UE waypoint(s) if it has.

NWDAF provides an identifier for each network slice instance and its associated load level information. Several network functions are using this NWDAF, such as the Policy Control Function (PCF) and the NSSF (Network Slice Selection Function). The PCF may use that data

in its decision policies, and the NSSF may use the load level information provided by NWDAF for slice selection. This information can be used for optimization and seamless transition of the handover process while ensuring better quality of service with direct impact on QoE improvement. Data related to UE behaviour can be also useful in case of analytics to introduce tailored services taking into account the habits and preferences of the client.

3.3.3 Use case related sensors for Slice, Resource and application

The eHealth requires sensors to retrieve traffic-related information from the several network points and calculate the QoS (traffic-related) metrics described in section 3.3.4. Therefore, network probes are required to provide traffic information at the slice level. To support the foreseen slice management dynamicity, the network probes should be dynamically instantiated, reconfigured, removed, etc., according to the service and slice needs.

As described in [18], the following table lists the parameters that should be taken into account for the use-case.

Table 4 Use case related parameters

Service Monitor	Video quality: Frame rate; Frame resolution Bitrate/Encoding Rate Service availability/Coverage level
Slice Monitor	QoS: monitor latency, jitter, packet loss, packet error rate, availability time of each network connectivity link (network availability), and each components. Security: monitor traffics (traffic flows, flash events, traffic patterns, etc.), monitor equipment behaviours (registration, activities, etc.) for anomaly detection, offline data analytics for further breach detection/analysis. Fault and Performance: monitor network performance variations and trends (QoS monitoring targets above), etc.
Resource Monitor	RAN resources: Radio link quality, signal strength, traffic congestion in current cell where the information is useful to make a decision when to start a handover process (to new Base Station) for better link/traffic. Channel Quality Indicator (CQI) to enforce a new resource allocation policy or change the content quality to optimise video streaming and QoE. Topology/UE mobility. Coverage: Number of active Base Stations and overall coverage area, in case of events reducing the coverage level, some actions to consider include increasing the number of active BSs or signal strength adjustment for each BS to achieve the required coverage level. Fault and Performance: monitor hardware equipment (detecting faults), interference (corrective actions: frequency reallocations,

	antenna/radio adjustment, coordination between Base Stations), network capacity (rescheduling algorithms to reduce/avoid congestion, etc.), etc.

3.3.4 QoS metrics

QoS KPIs which have a direct impact on QoE should be measured. Mobility interruption time is defined as the time span during which a UE cannot exchange UP packets with any BS during transitions. It can be regarded as intra system handover interruption time.

For video quality , it is good to monitor information related to video from RTP header: videoCodec, videoFrameRate, videoRTPPayloadType and videoDestPort.

Threshold for intersystem interruption time from LTE to 3G should be 500 ms.

Threshold for intersystem cell reselection with redirection interruption time from LTE to 3G should be 3s.

Handover success rate should not be worse than 98%

Thresholds for interruption time in Non-Standalone Architecture (NSA) option should be 100 ms.

Threshold for intersystem cell reselection with redirection interruption time in NSA option from LTE-NR to LTE should be 1ms. In this option , it is almost transparent because of dual connectivity.

Handover success rate should not be worse than 98%. Threshold for interruption time in Standalone Architecture (SA) option should be 5 ms

Threshold for intersystem cell reselection with redirection interruption time in SA option from NR to LTE requires additional tests to confirm threshold because the value offered by sellers on this KPI range from 90 ms to 1000 ms.

In [21] introduced the following plot for bandwidth and latency.

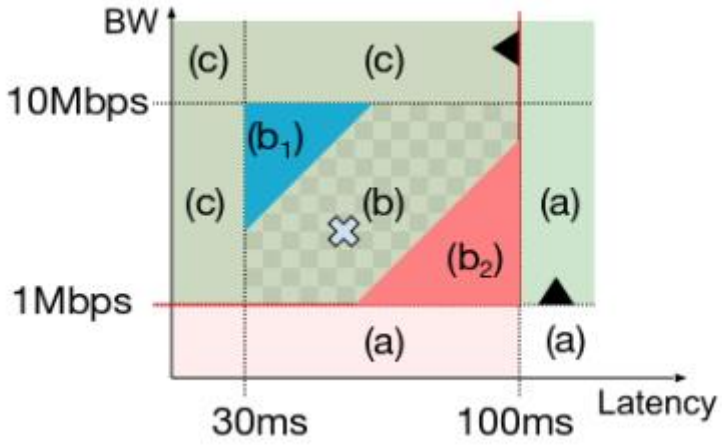

Figure 21 2D scatter plot for bandwidth and latency

Region (a) is considered as an unacceptable. It means that NSP knows that if the QoS falls within this area the QoE will be poor. Region (b) is considered as optimal QoS region. The customer presumably will be happy with the QoE if the provided QoS falls within this area. In region (c) the customer should be happy because he is getting better service than the one requested.

3.3.5 QoE metrics

The challenge with radio is that channel conditions vary so it is difficult to allocate a quantum of resources which will provide a deterministic bit rate. Resource blocks in a 20Mhz channel bandwidth will use different modulation schemes based on the quality of the channel. One approach is to use adaptive feedback and learning based on the measured CQI (channel quality indicator) and to adapt the codec and uplink throughput based on the capacity of the channel. CQI and other parameters which affect the selected modulation schema could be helpful although not many are exposed in the UE and so are not always available to the codec. This approach has the potential to optimise the image received although it has to been validated yet in the field.

QoE Sensors:

1. Video Sensor, IoT and a QoS Sensors (CLI speedtest or iperf or ditg are useful for this) are required
2. Collect video specific and flow level metadata and metrics.
3. In terms of latency, jitter and packet loss rate, Cisco TelePresence could be a reference point:
4. 30ms to 100ms latency end-to-end.
5. 10 ms peak-to-peak jitter
6. 0.03 to 0.05% random packet loss

QoE perceived by user is not only based on communication systems (related to QoS) but also affected by influence factors which can be related to ambulance mouvement.

Relevant features for QoE assessment:

Typical telemedicine applications include the transmission of:

- basic patient information,
- video images ,
- patient interviews and examinations,
- consultations with medical specialists,

Influence factors on the QoE for telemedicine:

- Human influence factors: Emotional state (tiredness, stress, eyestrain)
- System Influence factors: coding, transmission, display

Influence factors context:

- Clinical factors : lesion subtleness, emergency care
- Requirements : real time; location (moving ambulance)
- Medical data :
    - data types : signal, images, video, audio

Audio as a medical information is very important. Key factors are packet loss rate, jitter and delay. The impact of each individual or combined factors lead to blocking, blurriness or even blackouts with different levels of quality degradation of video for example. Delay has a direct influence on user perception. The video QoE degrades almost linearly as jitter increases.

4 Monitoring framework design aspects

In the context of Slicenet, the role of data monitoring is to provide the raw material to the QoE sensors. The Monitoring Framework (MF) has a twofold role: first, it tries to address the challenges of maintaining a history of massive-scale monitoring data; second, it provides the appropriate APIs to the QoE sensors in order for the latter to gain access to both raw monitoring data from the Control Plane (CP) and complex data as a result of further processing. In essence, a MF is the mediator between the CP laying at its southbound and the QoE sensors running at its northbound. In what follows, we array the design challenges and principles of a RAN MF that has the above stated role in the context of SliceNet.

4.1 Design challenges

4.1.1 Seamless Data Flow

Data flow poses the greatest challenge for a MF due to a series of reasons, namely, the massive scale; the high *frequency* and the different levels of *granularity* of the continuously produced monitoring data, the low time-scales (e.g., typically 1 ms during a Transmission Time Interval (TTI)); and the different control and management applications that must exchange raw or complex data. The focus has to be put on the underlying *data store* that maintains seamless flows of data between the network controller and applications (referred to as “apps”, acting as either data monitoring or data processing applications) like the QoE sensors, which use online or historic data.

4.1.2 Cost of monitoring

The cost of monitoring is expected to be higher compared to 4G networks due to the reasons above. In addition, it is not always clear which *party* (i.e., a DSP or NSP, or even the verticals themselves) has the responsibility and/or the ability to take over the cost of data gathering and maintenance. The main design challenge here refers to enabling to share costs and responsibility to a desired extend via different views, levels of granularity and aggregation of shared data after rules and policies defined by the different parties. The underlying incentive for the different parties in order to share monitoring data in a common MF is a mutually beneficial QoE monitoring and -as a result- slice performance at the cost of compromising some contradicting business interests (monitoring costs, adopting their privacy and security policies, etc.). Better QoE monitoring means a better resource utilization for all parties, which translates to minimizing operational costs and minimizing the impact of slice SLA violations.

4.1.3 Monitoring APIs

The design of a RAN MF must allow *flexibility*, providing the different parties with the proper set of Application Program Interfaces (APIs) and abstraction layers, which *hide technical aspects* and allow *filtering* and *aggregation* operations on data. In further, the flexible APIs can support heterogeneous underlying data store systems. Proper API definitions must encourage the development of innovative operations and facilitate the set-up of rules and *access-limitation* policies against harming the collected data and, therefore, the process of QoE sensing.

4.1.4 Scalability and other challenges

The MF components related to the underlying data store must be able to scale with respect to data volumes and the (high) interaction frequency with monitoring and processing apps. Bandwidth-guarantee and a fail-safe disaster recovery are also important challenges. Beyond the realm of its core design, the MF can leverage more than one data store endpoints (e.g., in the cloud) to increase availability, usability, bandwidth and fail-safe disaster recovery. Such a distributed data store can scale on demand with network requirements to optimize monitoring costs and performance, while this remains transparent to the monitoring applications consuming or processing the monitoring data.

In addition, the MF design must have features for analysing the resource usage by the multiple slices and the overall latency of monitoring requests. Last, the design has to allow accountability metadata regarding SLA violations, and be able to *fire alarms* on time for the purposes of (quasi-) real-time network control decisions.

4.2 MF Architecture Design

Figure 22 below presents our envisioned MF design and its interfaces. It provides a high-level view of the interaction of the MF modules to the CP in the southbound and the northbound monitoring apps. At the southbound, a CP producer API writes historic data measurements and statistics to the data store, which as shown in the figure can be a cluster to address the scalability challenges explained above. Measurements are in an adjustable granularity. Nevertheless, this is preferably at the highest possible granularity to allow further app processing. On the northbound, lie monitoring apps that consume data, passing their requests via a *"FILTER"* module that selects only the desired data or desired aggregating results during a selection process that takes place on a per user, slice, and cell basis. In further, filtering enables the control of access, create, read, update and deletion operations to the data store. Consumer apps can also retrieve data from the "*Message Bus"* when a message is published on a subscribed channel or when a data store watcher fires after some rule, e.g. due to a metric threshold breach. It is important to note. As previously stated, the MF APIs in the figure allow a diverse set of underlying frameworks like different data store frameworks to be used, by hiding their heterogeneity via converting app requests for data like QoE sensor requests, to meaningful requests for the underlying data store module.

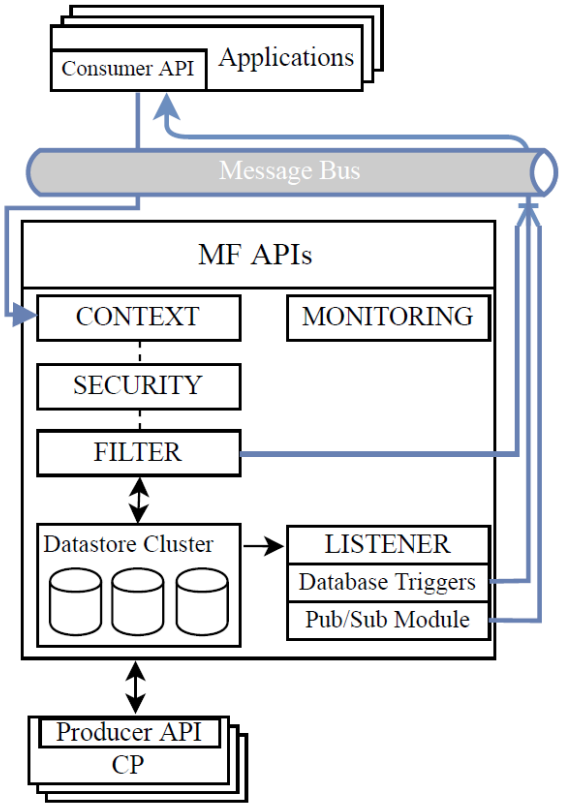

Figure 22 Monitoring Framework Design and Interfaces

The Message Bus, in particular, is a bidirectional communication means between data producers and consumers. Considering the challenge of high data and communication frequency, the Message Bus must address the high requirements of massive network requests. This translates to storing measurements with millisecond sensitivity and with very little tolerance to data losses such as in the case of QoE sensors crafted particularly for seamless ambulance handovering (see Sec. 3.3.2) in the UC of eHealth. Also, it should satisfy the data consuming apps with a stable performance. This design approach is best suited by separating production from consumption requests to different Message Buses, which can be, thus, configured after their usage needs.

As previously stated, facilitating different frameworks with a Publish/Subscribe (Pub/Sub) module or with database watchers can be handled by the MF APIs via different channels to lower the stress over the Message Bus. Evidently, CP measurements must be sent over a direct communication channel with a *higher write performance* to the data store. In further, the main monitoring publisher app can publish data in to topics where different clients apps can fetch them in different rates, and rewrite data on their *own channels* to output information. However, standard message queue brokers cannot be used for this purpose; once a piece of data gets published to a topic, then it goes to every client even, including clients that would like to receive this piece of data at a later time. Ignoring some input packets in the channel by the client side is a suboptimal optimal choice, as it would create a unnecessary heavy server load due to the data that would still flow to all clients. A proper design solution should allow client apps to make requests and read purely on different frequencies using publish-subscribe messaging system such as Apache KAFKA.

4.3 Monitoring in Support of Cognitive Slice Management

The proposed MF provides the data infrastructure to enable SliceNet's Cognitive Slice Management (as part of AI driven network management and, more generally, AIOps). It embraces the concept of a Data Lake, providing a repository that can store both structured and unstructured data at scale. QoE awareness requires inherent flexibility and dynamicity to support different vertical applications, which means that we do not always know in advance what data needs to be collected and stored to allow QoE-aware management. Relevant data may come from multiple data source, such as DP/CP sensors (resource telemetry, traffic metrics and topology), Slice and Sub-Slice metrics (slice topology, aggregation and analytics), vertical inputs, and even external sources. Some data ingestion steps (such as information extraction, verification, and transformation) may depend on the specific infrastructure and/or Slice characteristics; thus, even the schemas may be defined only at analysis time. The same flexibility applies to data analysis (such as contextualization, classification, and prediction), where different methods may be required to support different Slice classes. More generally, SliceNet's vertical-driven approach requires that the monitoring data path may be tailored to support each Slice, combining multiple data sources, various data preparations, and data analytics.

As described in the next sections, this MF can be utilized to implement QoE sensors, which convert vertically agnostic QoS metrics into vertically perceived QoE. Moreover, the framework supports the application of machine-learning techniques, allowing integration of data preparation and analytics as part of the monitoring data path as well as longer-term data storage in support of learning new models and model training. It should be appreciated however, that the framework enables other rich and flexible sensors that provide critical CP information in support of Slice management. For example, a predicted signal may be used as a sensor to proactively trigger auto-scaling, avoiding service degradation. Such signals are essential for ensuring QoE aspects that cannot be measured directly (e.g., availability or security).

5 QoE Sensing

QoS refers to deterministic network behavior, so that data can be transported for example with latency, bandwidth, packet loss… on the other hand, QoE is a subjective measure that involves human dimensions. It links the user's perception, expectations, and experience with application and network performance. In the context of slicing, measuring perceived QoE is a challenging problem for verticals because it is costly and complex due to the human involvement in the process.

The challenging issue of the subjective measurement is to predict it from the objective measurements; in other words predict QoE from a given set of QoS parameters.

In this context, for the QoE metrics, we need to identify **slice resources** and their exposed **QoS metrics**. These resources are of different "types": e.g. VNFs that are explicitly part of the Slice descriptor, identified as a resource of the Slice, whereas a Flow is a service enabled by the end-to-end Slice.

We define also the term "**ObservationPoints**" that represents the points/resources from where QoS metrics are collected. E.g. an end-to-end user flow can be monitored at different observation points.

To meet the needs of the verticals in terms of QoE, we need to know (1) data that the vertical wishes to supervise, (2) metrics to collect from NSPs and specifically from the ObservationPoints and (3) the aggregation techniques and the operations that define the relationship between the QoE and QoS metrics. In this section we will describe the requirement for QoE sensing in terms of metadata and aggregation and the proposed QoE monitoring architecture.

5.1 Requirement for QoE sensing: metadata and aggregate

This section describes the required data that are needed to conceive the QoE sensors:

- raw data (or the telemetry): these are the metrics that are directly collected from the observation points. QoS sensor samples the state of the monitored resource (eg. a flow) and reports values of each metric associated to the monitored resource
- metadata: describes the data itself, different attributes are required to describe the data like timestamp, name, ID...
- Aggregated metrics: these are metrics that are obtained from the raw metrics e.g. aggregating network related parameters (example bit rate, packet loss rate, jitter, etc.) to predict QoE [11,12]

5.1.1 Meta-data for QoE metric

The vertical can request to estimate QoE metric for a specific UE, set of UEs or the entire traffic crossing his slice. The metadata describing resources for the purposes of QoE sensing will include the following elements:

- Timestamp: temporal information regarding the time when the QoE value is computed (by estimation or prediction) from the QoS metrics.
- Name: metric name
- Value: metric value
- SliceID: the vertical slice ID

- MetricType: represents the metric type (e.g. counter, gauge…)
- ObservationPoint: locations of QoS sensors
- etc.

5.1.2 Aggregated data for QoS metrics

Once data is collected at the NSP level, two options are possible. The first one consists in pulling raw metric data and then applying data aggregation using specific operator (e.g. sum, average…) on them. In this option, the QoE sensors take stored raw metric values as input and computes QoE indicators using a model mapping QoE to QoS metrics influencing it. This model can represent a specific function defining the relationship between QoS and QoE or a ML model that is already trained on a specific datasets. The second option refers to the case where the data aggregation is done when pulling data from the monitoring system in the NSP level.

For metrics aggregation, we distinguish three types of aggregation:

- Spatial data aggregation: is the aggregation of all data points for a group of resources over a specified period (the granularity). The spatial domain consists of a set of disjoint resources e.g. aggregating data from all VNFs composing the slice.
- Temporal data aggregation: the aggregation imparts the temporality to the data by maintaining the time of sensing along with the sensed data.
- Data aggregation of heterogeneous metrics: the data aggregation is conducted on data coming from different QoS sensors.

These aggregation techniques can be combined between them to meet the needs of QoE sensor e.g. spatio-temporal data aggregation.

5.2 QoE monitoring architecture

The QoE monitoring architecture aims to enable the dynamic deployment of QoE sensors and to retrieve data collected by these sensors. Figure 23 provides a view on components responsible to achieve the sensing of the QoE metrics expected from the vertical requirement. The QoE monitoring architecture will expose a northbound interface toward the QoE optimizer module to deploy new QoE sensors and to retrieve the collected QoE metrics.

On the other hand, the architecture will provide a set of components responsible to interact with the underlying infrastructure (NSP) in order to ensure the name resolution of parameters (e.g. observation point to IP address) coming from the northbound interface and to retrieve and aggregate QoS metrics composing the QoE metric.

In addition to that, the architecture will host the QoE sensors that will, periodically, aggregate the QoS data and store them in a shared database.

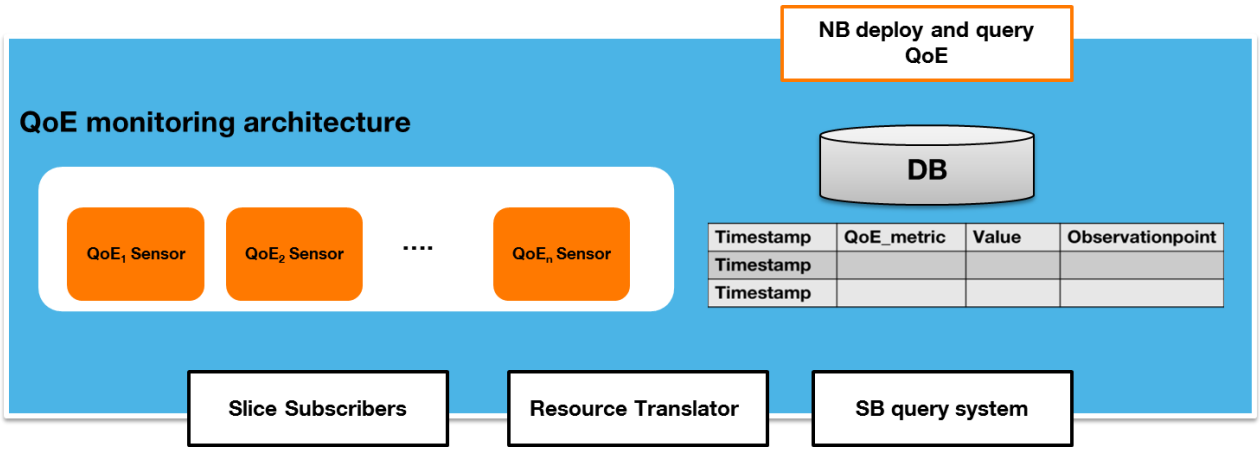

Figure 23 QoE Monitoring architecture

The architecture components will be described in the following subsections.

5.2.1 Resource Translator

The resource translator component have two main roles: the first one is to retrieve the identifiers of network slice resources (NSR) from the NSP hosting the slice based on the sliceID. The second role is to map the obtained NSR to the corresponding records. These records represent for example the end point address of the network slice element at which metric should be retrieved (observationpoints). This information can be obtained from the resource inventory running at the each NSP. The resource translator component is not mandatory in case that the observationpoints are already translated.

5.2.2 Slice Subscribers

The Slice subscribers' component contains information about UEs attached to the Slice. This information is needed to supervise the E2E traffic of UEs. Note that the QoE sensor can collect information of a single UE, or a set of UEs or all UEs connected to the slice. To achieve these needs, the slice subscribers are expected to return the following information:

- IMSI (International Mobile Subscriber Identity): to be able to collect flow metric from the RAN segment
- State: the state of the UE if it is connected or not to the slice
- IP: address of the UE in order to be able to collect flow metric from the Core segment
- SAP (service access point): the point (e.g. RAN) from which UE is connected to the Slice

5.2.3 Database

The database component stores all the collected/estimated QoE metrics, along with their corresponding "time series". The QoE metrics are generated and collected from QoE sensors. Each element of the database will contain at least:

- Timestamp: the creation time of the QoE entry in the DB.
- QoE_metric: the metric name
- Value: the QoE predicted value
- Observationpoints: locations of QoS sensors
- UEs: list of supervised UEs

5.2.4 Southbound API

The Southbound query system queries the monitoring modules at the network service provider (NSP) level to get QoS values. To perform this query, two parameters are needed:

- ObservationPoint: indicate the location where QoS metrics must be collected
- UEs: all UEs connected to the slice, a subset of UEs (e.g. two UEs for point to point flows) or a single UE (i.e. all flows concerning this UE)

5.2.5 QoE sensor

QoE sensors are responsible for generating QoE metrics using the collected QoS values. This can be assured through two ways:

1. using a specific function that defines the relationship between QoS and QoE (see Figure 24). In this case, sensors dependencies should be, first, resolved using the resource translator module (e.g. map SAP from which UE is connected to IP address) and the slice subscribe module (e.g. mapping UE to IMSI) (Step 1, Figure 24). Once dependencies are solved, the SB query system queries the monitoring system of the NSP (Step 2, Figure 24) and the specific function defining relationship between QoS and QoE is executed (Step 3, Figure 24). The calculated QoE value is, then, stored in the database (Step 4, Figure 24). These steps are repeated for each sampling time.

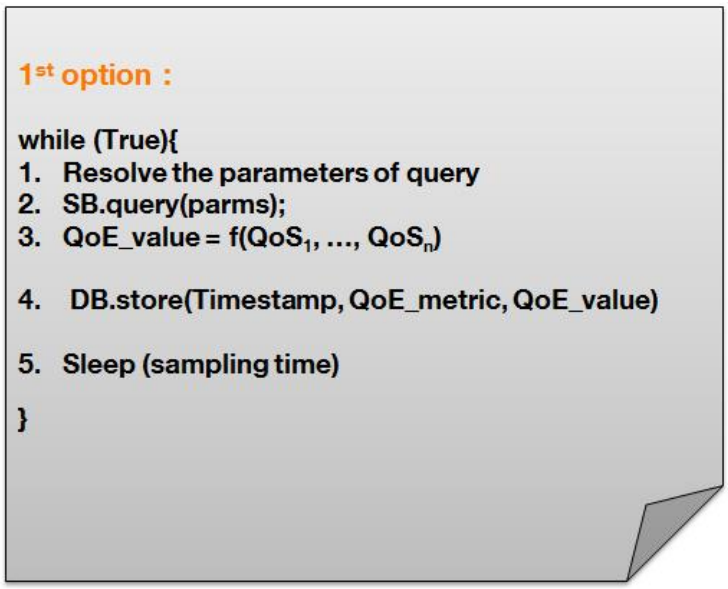

Figure 24 QoE sensor (option 1)

2. using a predictive model for QoE based on classification/estimation that learns through training (see Figure 25). Like the above case, the first step is to solve sensors dependencies (Step 1, Figure 25) and the second step is to query the NSP monitoring system (Step 2, Figure 25). Once QoS values are retrieved, the model is called to predict or estimate QoE values or to predict the QoE class (Step 3, Figure 25). The predicted QoE value is stored in the database (Step 4, Figure 25). These steps are repeated for each sampling time.

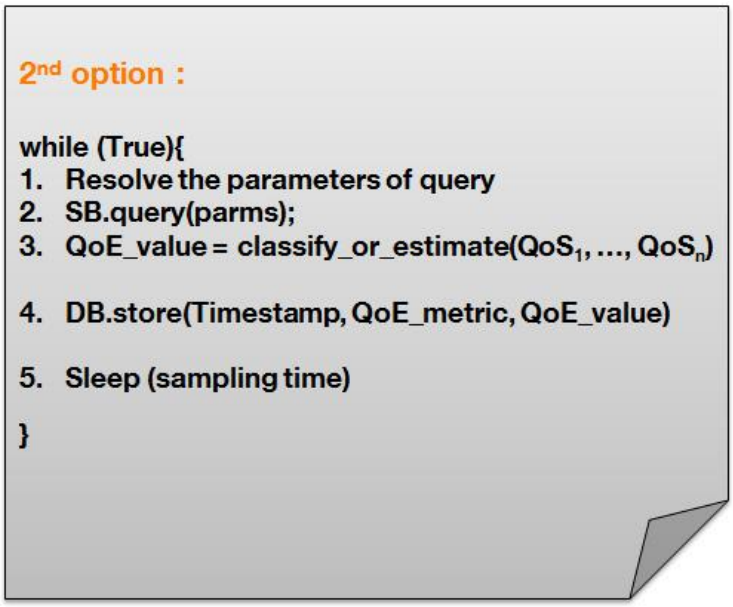

Figure 25 QoE sensor (option 2)

Figure 26 gives more details on how to generate the predictive model. If the vertical provides exhaustive datasets (enough data to train), then this data will be used for training and the QoE sensor will be generated (step 2) and deployed (step3). But, if the vertical provides limited datasets or does not provide any data, data is generated using simulation (step 1) in order to be used for training (step 2). Then, the QoE sensor will be generated (step 2) and deployed (step 3). These steps will be more detailed in section 6.

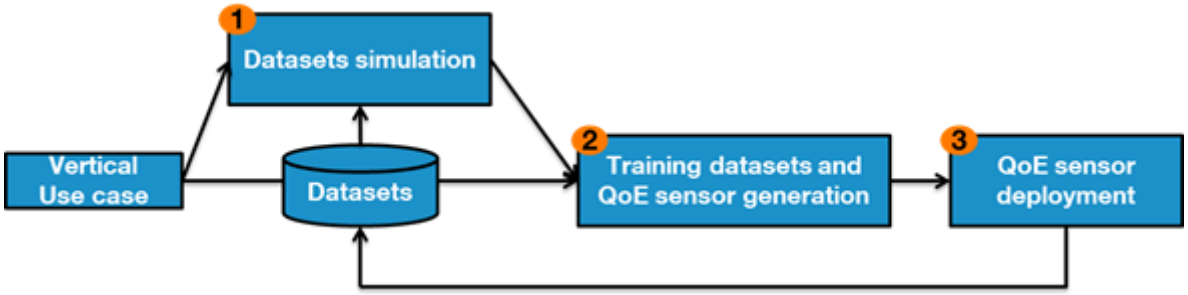

Figure 26 Process to train, produce and deploy QoE sensor

5.2.6 Northbound API

The Northbound API enables the automatic deployment of QoE sensor by specifying the parameters below in the deployment request:

- model/function: represents the relationship between QoE and QoS metrics
- QoS: the list of QoS metrics
- UEs: the list of UEs to be supervised
- slice ID: the vertical slice ID
- The time window for data collection
- The observationspoints: QoS sensors from which data is collected

As depicted in Figure 27, QoE sensor deployment requests invokes the creation of a new QoE sensor instance packaging the prediction/function module (steps 1, 2). As described in

the sections above, the deployed sensor interacts with the slice subscribers and the resource translator modules to resolve identities of slice resources and the UEs (step 3). The SB query system retrieves QoS data from NSPs (step 4) and the predicted QoE value will be stored in the database (step 5).

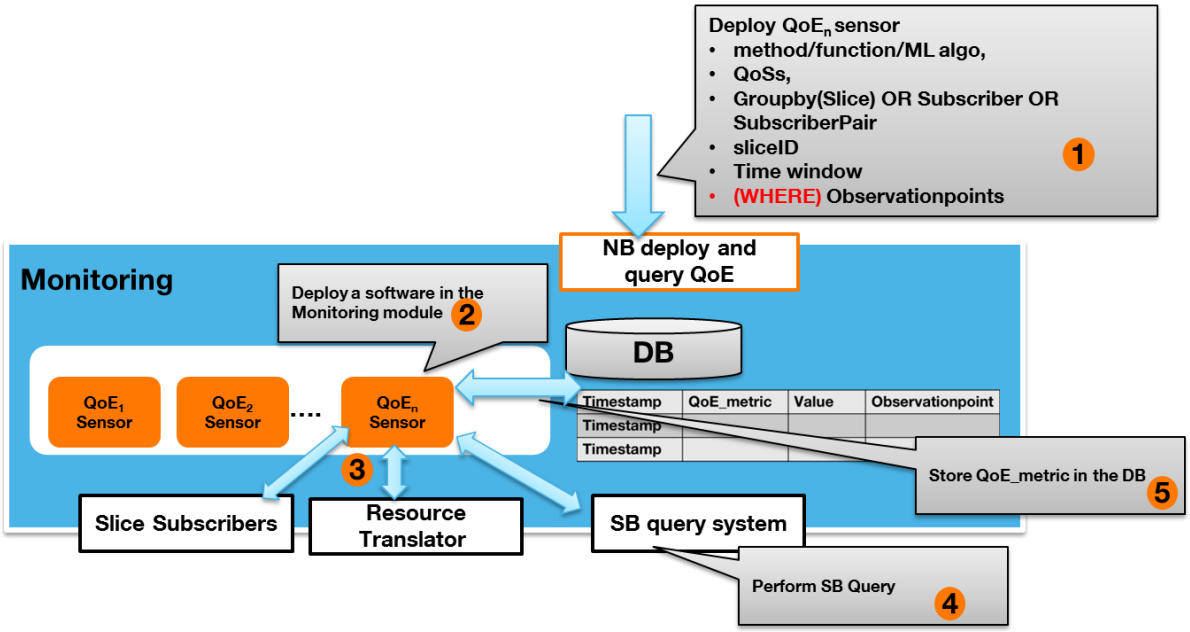

Figure 27 QoE sensor deployment

The Northbound API allows also retrieving the values of QoE metric stored in the database, as depicted in figure below. This API will allow extracting QoE values using specific operations like average, sum… and performing aggregation techniques on QoE metrics.

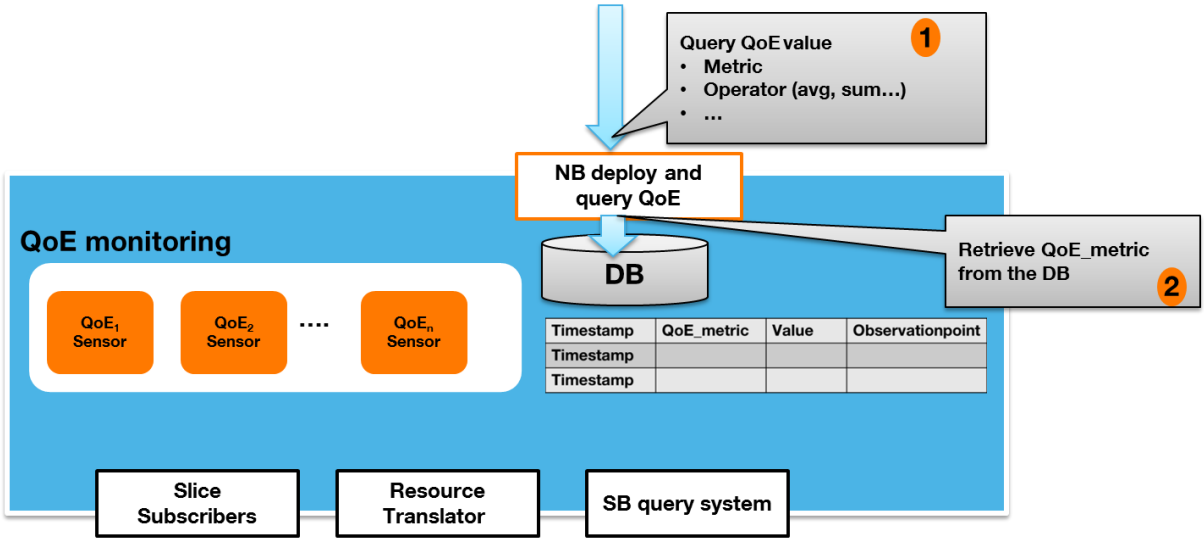

Figure 28 Retrieve QoE values

5.2.7 Architecture design

Figure below depicts an example of an implementation of the QoE monitoring architecture on the RAN segment. Data collected from the RAN through FlexRan is stored by the QoS monitoring tool at the NSP level. The QoE sensors are instantiated as applications deployed by the Monitoring Framework (MF detailed in section 4) i.e. App 1, App 2…. The "NB deploy and query QoE" interface of the QoE architecture, is mapped to the MF northbound API to enable the deployment of Apps and to retrieve data stored in the MF. Note that each App, representing a QoE sensor, will use the interface provided by FlexRan to perform slice resources translation and subscribers mapping. Then it will use these identifiers to query needed QoS metrics from MF database to perform QoE estimation or prediction. Finally, these applications store the predicted QoE value in the MF database. Figure 29 shows the mapping of the QoE monitoring components on the MF components through the different arrows.

Arrow 1: represents the mapping of QoE sensor on the MF App.

Arrow 2 and 3: represent the mapping of the "slice subscribers" and "resource translator" on the link between MF App and FlexRan.

Arrow 4: represents the mapping between the "SB query system" on the link between MF App and the MF database.

Arrow 5: represents the mapping of the QoE monitoring DB on the MF DB.

Arrow 6: represents the mapping of the "NB deploy and query QoE" component on the MF northbound API.

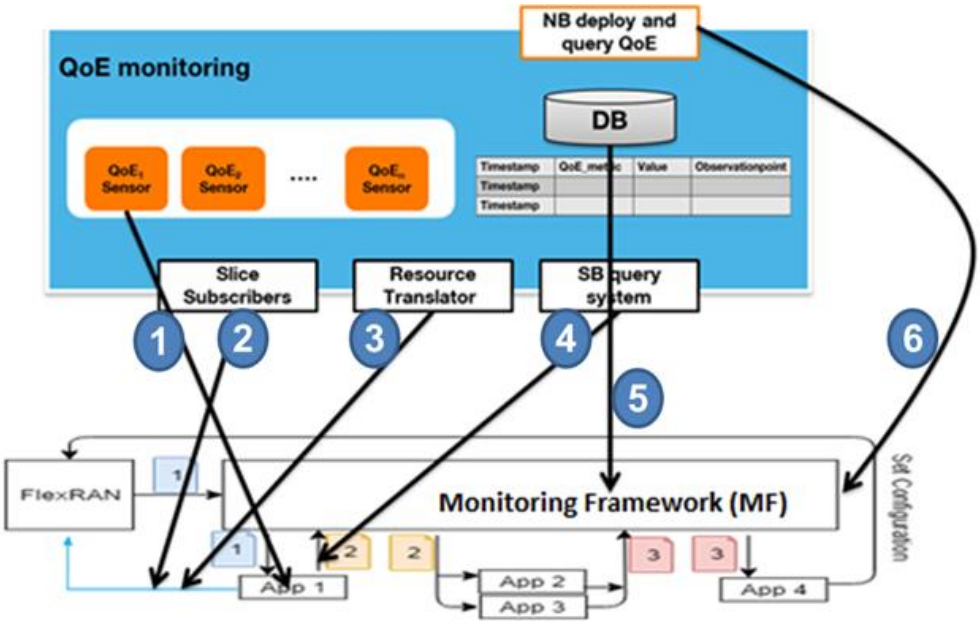

Figure 29 Mapping of the QoE monitoring architecture to the MF

6 Machine learning for “intelligent sensor”

In this section, we will focus on the option where the QoE sensor is based on machine learning techniques in order to predict the QoE from QoS indicators. First, we present our view of the different steps to follow for the training and the prediction phases. Second, we present how the nature of QoS indicators can be defined from a mathematical perspective since all the steps of the training phase depend of this information. Third, useful pre-processing techniques for the QoS indicators are proposed in order to prepare them for the training. The last part of this section will present ML models that can be useful for the QoE sensor. The first model aims to detect future anomalies in RANs. By predicting in advance the possible anomalies that may occur in the RAN, not only the QoS of the network will improve but also the QoE perceived by the end user. For this model, a code is provided in GitHub. The code details the training and prediction phases’ implementation with R language. It also allows testing the model through a simulated data since real data could not be shared. The second model aims to detect performance degradation of the VNFs by predicting if VM/VNF is suffering from noise or if it is stressed by applications running on it. Supervised learning algorithm has been used to identify the noisy neighbour phenomenon.

Since data are not yet provided by different SliceNet verticals, the first model has been evaluated using real data extracted from internal networks of Orange France. The second model has been evaluated using a simulated environment.

6.1 Training and prediction cycle

The QoE sensor aims to guarantee a good QoE perceived by the end users by providing a top quality of services. Apart from QoS indicators, the verticals may have other types of data such as pictures, videos and audios for the eHealth use case or weather information for the Smart city use case. One possibility for the QoE sensor is to deal with these different sources of data in order to predict the QoE. However, since some of the information may be confidential such as the patient radiographies or they may not explicitly reflect information related to the network, we propose to work with QoS information. For this reason, the verticals should consider to extract QoS indicators from their raw data. They may also consider simulating other datasets if the raw data does not cover some of the possibilities such as the death of one patient because of service disruption. Once the training set is ready, the QoE sensor’s objective is to predict the QoE. In this section, we focus on the option where the prediction is achieved using machine learning. The ML problem can be defined as follows:

1. A classification problem when the QoE that we aim to predict is expressed as a qualitative variable
2. A regression problem when the QoE is expressed as a quantitative variable
3. A multi-class problem when the objective of the ML algorithm is to predict not only one QoE indicator but a set of QoE indicators at once.
4. A forecasting problem when the objective is to forecast future QoE.

The result will be a ML model that is ready for exploitation. Once we have a new observation, the role of this ML model is to predict the QoE from the observed QoS indicators. In order to benefit from each prediction, this latter could be used to enhance the training data for future uses. To do so, each prediction should be first validated by the verticals. The validation could be achieved by comparing the received data with the sent

data. By adding these new observations to the training set, the ML model could be updated either dynamically by using incremental classification algorithms or statically by planning periodic episodes of learning.

Before applying classification, a pre-processing step of the data should not be neglected. Actually, if there is much irrelevant, redundant information or noisy and unreliable data, then knowledge discovery during the training phase may be misled. Hence, data pre-processing is one of the most critical steps in the QoE sensor which deals with the preparation and the transformation of the QoS indicators in order to improve the efficiency of the mining process. Pre-processing includes data cleaning, data transformation and data reduction. It could also englobe an unsupervised training step in order to discover the hidden patterns in the QoS indicators. Through an unsupervised learning step (clustering or co-clustering), clusters having the same behaviour may be extracted and labelized by experts (with the help of the verticals). The product of data pre-processing is the final training set. The whole chains of the training and the prediction phases for the QoE sensor are presented in Figure 30.

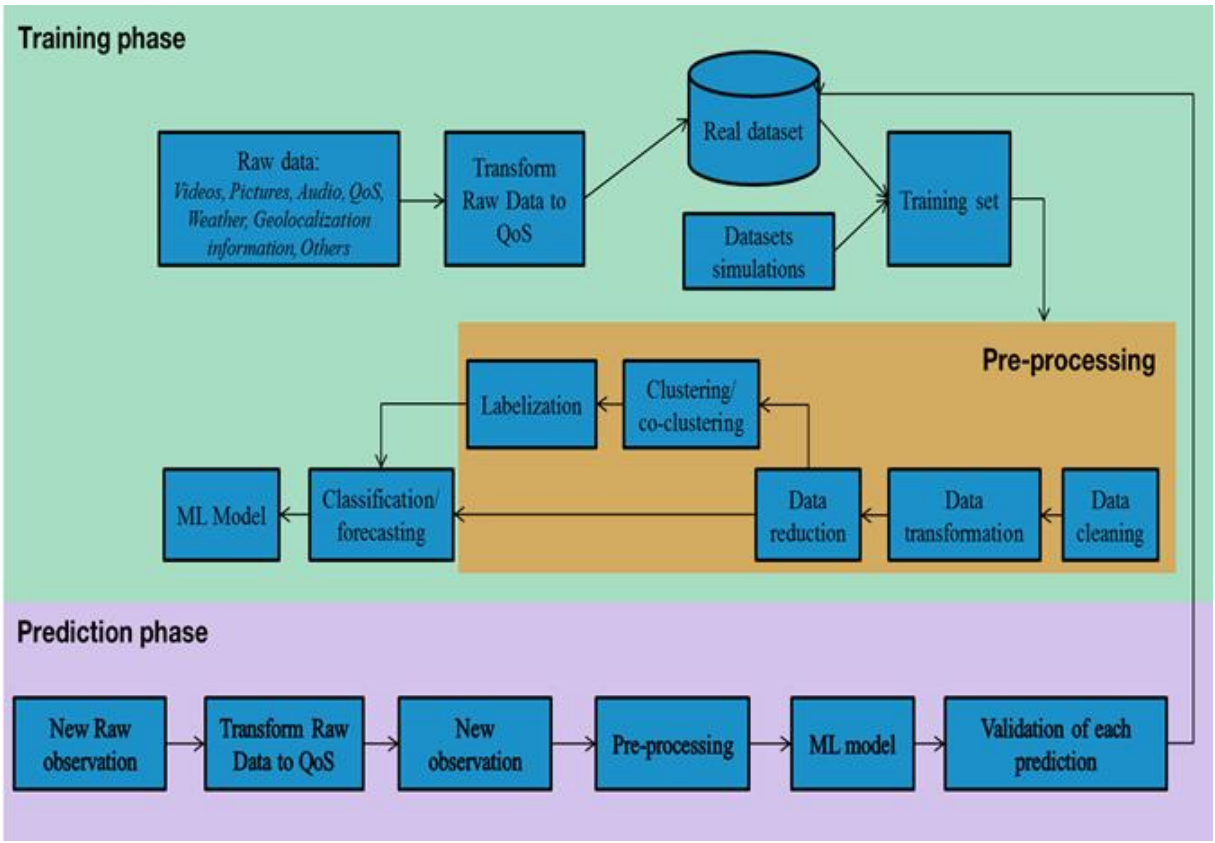

Figure 30 Training and prediction phases of the QoE sensor

6.2 QoS indicators from mathematical perspective

QoS indicators are measurements computed periodically from the network with different temporal granularities (daily, hourly or less). A QoS indicator can be expressed as a single numeric variable or as a set of measurements along a time interval. In this last case, they are commonly treated as multivariate data i.e. a vector, which implies that the different measurements for each QoS indicator are assumed to be independent. This assumption

simplifies the ML process but it is not realistic since the measurements are chronologically ordered and therefore they are correlated.

A second alternative consists to express the QoS indicators as time series. This alternative is useful especially when we aim to forecast the QoE. The most popular models are Auto-Regressive Moving Average models (ARMA, [1]) that relates the present value of a time series to past values and past prediction errors. Although these models have proven their efficiency with time series analysis, this latter have some limitations. First, in time series analysis, it is assumed that the data are observed at regular intervals of time. They are therefore not appropriate for irregularly spaced time series as the path is no longer a constant. A very common approach of modelling the time series with irregular spaced time is to convert them into regular time series using techniques such as interpolation, and then to model the resultant series using regular time series methodologies. Such methods risk resulting in higher bias/errors [2], [3].

In order to convey to these limitations, Functional data analysis is a third alternative. In this case, a QoS indicator can be considered as functional data [4]. According to [5], a functional random variable X is a random variable with values in an infinite dimensional space. Then, functional data represents a set of observations $\{X_1, \dots, X_N\}$ of X. The underlying model for X_i's is generally an i.i.d. sample of random variables drawn from the same distribution as X. A well accepted model for this type of data is to consider it as paths of a stochastic process $X = X(t);\ t \in T$ taking values in a Hilbert space H of functions defined on some set T. Generally, T represents an interval of time, of wavelengths or any other continuous subset of R. The advantage of considering the QoS indicators as functional data is that the discrete set of measurements of each QoS can be expressed in the form of a curve that represents the entire measured function as a single observation. Many advantages arise from this choice [6]. First, the generating models can be described by continuous smooth dynamics which may allow for accurate estimates of the parameters that have to be used in the analysis phase. Second, FDA methods allow the data noise reduction and also the treatment of missing data through curve smoothing. Third, by saying that a curve is smooth, we usually mean that it is differentiable to a certain degree, implying that a number of derivatives can be derived or estimated from the data. Such derivative information may reveal patterns in a (functional) dataset that address important research questions. Fourth, FDA methods do not make the assumption that the values observed at different times for a single subject are independent, which better corresponds to the functional nature of the QoS indicators. Finally, FDA is applicable to data with regular or even irregular time sampling schedules.

6.3 Pre-processing of QoS indicators

6.3.1 Data Cleaning

Generally, QoS indicators may contain some inconsistency, some incomplete information and some noise (such as errors, or outlier values which deviate from the expected. This can occur for a number of reasons: (1) Attributes of interest may not always be available, such as the anomaly behind an alarm; (2) some data may not be included simply because it was not considered important at the time of entry; (3) relevant data may not be recorded due to a misunderstanding, or because of probes malfunctions; (5) there may have been human or computer errors occurring at data entry; (6) errors in data transmission can also occur. There may be technology limitations, such as limited buffer size for coordinating synchronized data

transfer and consumption; (7) incorrect data may also result from inconsistencies in naming conventions or data codes used.

Hence, data that are considered as inconsistent with other records or redundant should be deleted. As for missing values, they can be filled in for the attribute by various methods described below:

- Ignore the tuple: This is usually done when the class label is missing in a classification context. This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.
- Use the attribute mean to fill in the missing value
- Use the attribute mean for all samples belonging to the same class as the given tuple.
- Use the most probable value to fill in the missing value. This may be determined with inference-based tools using a Bayesian formalism or decision tree induction.

6.3.2 Data transformation

In this step, the QoS indicators are modified and consolidated into a more appropriate form before training. Data transformation can involve the following:

- **Standardization:** Since the QoS indicators are measured at different scales, they do not contribute equally to the analysis. For example, a variable that ranges between 0 and 100 may outweigh a variable that ranges between 0 and 1. Transforming the data to comparable scales can prevent this problem. Typical data standardization procedures equalize the range and/or data variability:

  x_standardized = (x - mean) / standard_deviation

  Normalization can also be used in order to scale the data. Normalization needs to know the max and min value of the data which needs to be normalized:

  x_normalized = (x – min) / (max – min)

- **Aggregation:** In some uses cases, it may be interesting to change the granularity of the observed QoS indicators. For example, the hourly scales QoS indicators may be aggregated so as to compute daily or monthly values. This could be interesting for the analysis of the QoS at multiple granularities.
- **Imbalance treatment:** In some cases, the training dataset may have imbalanced labels because one class may be more frequent than others. When the imbalance is important, reducing this imbalance in the data seems inevitable. Hence, a Synthetic Minority Over-sampling Technique (SMOTE, [7]) can be applied. SMOTE is a well-known approach that allows over-sampling the minority class in a dataset. The majority class examples are also under-sampled, leading to a more balanced data. The over-sampling is achieved by generating "intelligent" copies of the minority class observations i.e. by artificially creating synthetic samples. The new examples are generated by using the nearest neighbors of these observations. It consists in perturbing one attribute at a time by a random amount within the difference to the neighboring instances. This approach effectively forces the decision region of the minority class to become more general.
- **Smoothing:** The main source of difficulty when dealing with functional data consists in the fact that these latter belong to an infinite-dimensional space, whereas in

practice, QoS indicators are generally observed at discrete time points and with some noise. Even when the sampling rate is very high, the points, technically, are not continuous. As a consequence, the first step in FDA is often the reconstruction of the functional form of data from discrete observations.

To do this, a curve is fit to the discrete observations, which approximates the continuous underlying process. The most common procedure is to consider that sample paths belong to a finite dimensional space spanned by some basis of functions. This latter is a system of functions specially chosen to be used as building blocks in order to represent a smooth curve. Each observed curve X_i $(1 \leq i \leq N)$ is assumed to be decomposed as a linear combination of these basis functions $\{\phi_r\}_{r=1,\ldots,M}$:

$$X_i(t) = \sum_{r=1}^{M} a_{ir}\phi_r(t), \quad t \in [0,T],$$

where {air}r=1,…,M are the basis expansion coefficients for the curve Xi and they determine the relative weights of each basis function in constructing the built curve.

The original data are then set aside, and the estimated curves are used for the rest of the analysis. Further processing of the curves is also possible, such as the taking of derivatives or performing transformations. Many basis of functions exist in literature such as “trigonometric functions”, “bsplines” or “wavelets” (see [4] for a detailed study). The choice of the basis as well as the number *M* of basis functions is quite subjective [4].

For example, if the sample paths of *X* are cyclical and periodic, “Fourier” basis could be a good choice. If the data are noisy, “bspline” basis could be more appropriate due to the optimal properties of cubic bspline functions. If the data displays discontinuities and/or rapid changes in behavior, wavelet basis can be used. Once the basis is chosen, the estimation of the equation above can be done by interpolation, if the sample curves are observed without error, or more generally by smoothing [4]. Ideally, the smoother should reflect or have features that match those of the data such as illustrated in figure 31. By observing the original data and the smoothed curves, we could see that after the smoothing, the form of the data changed which risks in losing the information in the original data. Hence, the choice of the smoothing technique is dependent upon the underlying behaviour of the data being analyzed.

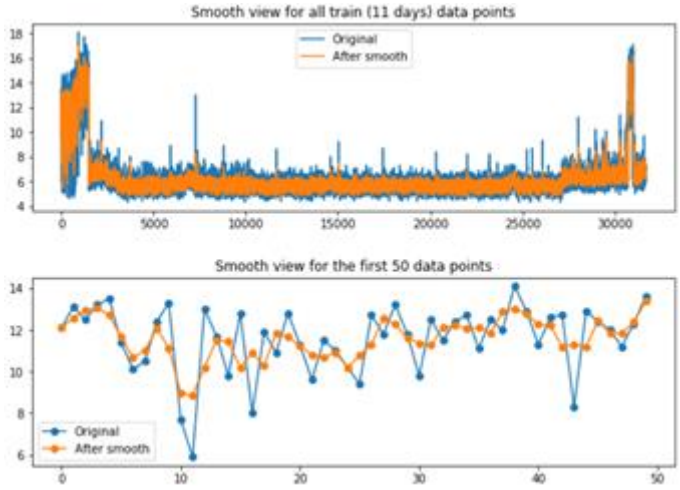

Figure 31 Illustration of the smoothed and the original time series

Regarding the choice of the number *M* of basis functions, the decision is often related to the bias-variance trade-off. A high number of basis functions will yield to a curve that is more faithful to the observed data (low bias) but that is often less smooth (high variance). However, using a small number of basis functions will produce a curve that places less importance on interpolating the discrete points (high bias) but more importance on smoothness (low variance). Thus, under-smoothing of the curves leaves in artefacts and variability (noise) that are not truly part of the process being observed. The resulting curve may thus represent perturbations in the observation process that do not pertain to the analysis being sought or the research question being asked. On the other hand, over-smoothing discards small scale, high-frequency behavioural data that may be part of the process we wish to observe and analyse. For instance, Figure 32 illustrates the observation of the DL traffic volume for one cell during one day where the data are extracted within 15 minutes. The raw data is, therefore, described by 96 discrete values. It also illustrates its smoothing with a B-spline basis. Different number of basis functions are considered *(M ϵ {5, 20, 80})* in order to illustrate the concept of under-sampling (when M = 5) and of over-sampling (when M = 80).

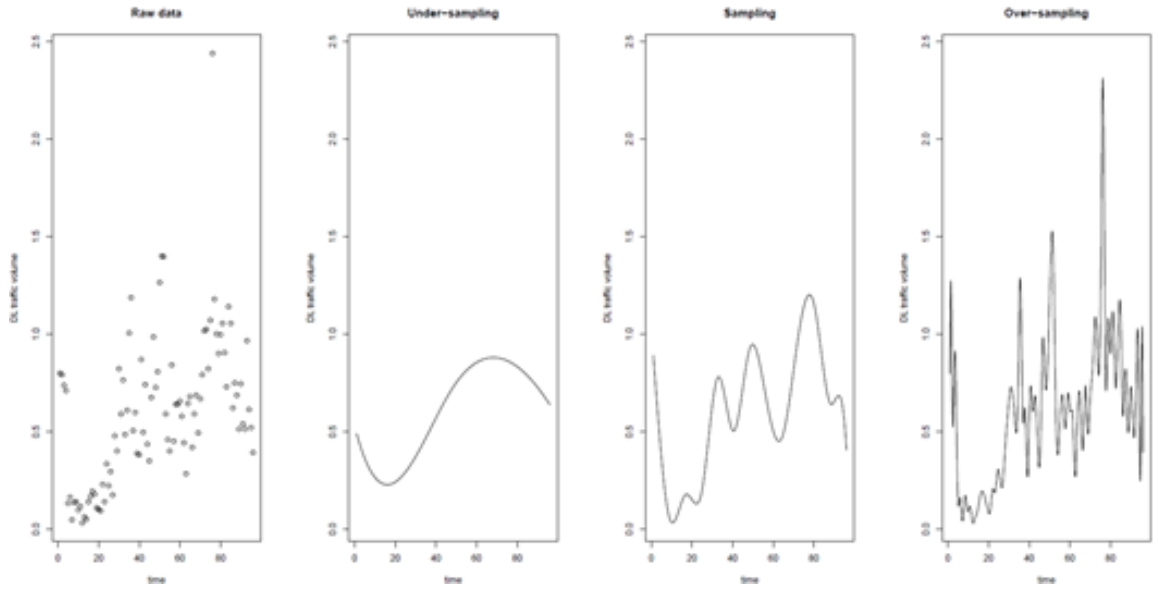

Figure 32 A daily observation of the DL tra_c volume KPI for one cell and its sampling with B-spline basis by using di_erent number of basis functions

- **Remove trend:** By applying the Augmented Dickey-Fuller test which is commonly used to test the stationary of a curve, we could observe if there is trend in the curves as illustrated below:

Table 5 Illustration of stationary test

```
# Stationary ADF test for the first 4 days' data

# We could say that the cpu time series in long term (14 days in our case),it's stationary;
# while in short term (for exmaple 4 days), it's not stationary.

stationary_test(cpu[:12000], message='(for the firt 4 days cpu_percent data)')
```

```
                ADF stationary test (for the firt 4 days cpu_percent data)

Augmented Dickey-Fuller test Statistic: -2.652489581715522
p-value: 0.08262658530342004
Critical Value:
        1% : -3.430896927350487
        5% : -2.86178171371544
        10% : -2.5668986591592824
Conclusion:
        It's not stationary
```

In case that the curve related to the QoS indicator is not stationary, we should test to remove the trend before enter it to the machine learning algorithm. There are multiple methods to remove the trend, one of the easiest and commonly used one is the differencing. It transforms the data point t by the difference between the data point t and the data point t-1. Illustrated in the Figure 33, we could observe that the curve is stationary after the differencing transformation.

$x_{(t)} = x_{(t)} - x_{(t-1)}$

Time series decomposition is also a technique that can be used in order to remove trends in the curves as illustrated in the right part of the Figure 33. The first image is the original data; the lower 3 images are respectively the plot of “trend”, “seasonal” and “residual” components. According to the decomposition method (addictive or multiplicative), we say that:

- Original curve = trend + seasonal + residual
- Original curve = trend * seasonal * residual.

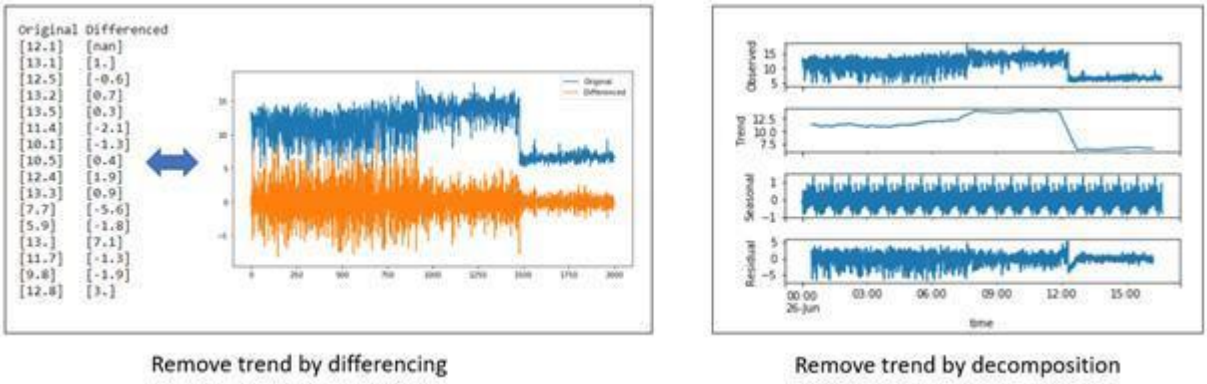

Figure 33 Illustration of Trend removal

6.3.3 Data reduction

Principal Component Analysis (PCA, [8]) is a key dimension reduction tool for multivariate data that is conducted when one wishes: a) to reduce the number of (possibly) correlated variables to a smaller number of uncorrelated variables and b) to reveal latent structure in the relations between variables. PCA finds the linear combinations of variables that highlight the principal modes of variation. It has been extended to functional data and termed Functional Principal Component Analysis (FPCA, [4]) since from the set of functional data, one can be interested in optimal representation of curves into a functional space of reduced dimension. FPCA consists in computing the principal components C^s, sometimes referred to as scores, and principal eigen-functions f^s of the Karhunen-Loeve expansion:

$$X(t) = \mu(t) + \sum_{s=1}^{S} C^s f^s(t).$$

where μ(t) = E(X(t)) is the mean function.

FPCA is widely used in FDA. This is partly because, under mild assumptions, the underlying stochastic process can be expressed as a countable sequence of uncorrelated random variables, the functional principal components (FPCs) or scores, which in many practical applications are truncated to a finite vector C. Then the tools of multivariate data analysis can be readily applied to the resulting random vector of scores, thus accomplishing the goal of dimension reduction. Besides, FPCA offers a practical interest for interpretation and data presentation through graphics and it facilitates the construction of parametric models that will be more parsimonious.

A common problem to FPCA is the choice of the number of principal components S needed for the approximation of the full Karhunen-Loeve expansion, which gives the best trade-off between bias and variance. In multivariate PCA, several procedures are routinely applied such as the scree plot or the fraction of variance explained by the first few principal components. These procedures can be directly extended to the functional setting.

In order to choose the best number of PCs, they use less subjective criteria such as pseudo-versions of Bayesian Information Criterion (BIC) and of Akaike Information Criterion (AIC).

6.3.4 Advanced pre-processing using co-clustering

In order to guarantee a top quality of experience, QoS is continuously measured. As a consequence, the amount of QoS indicators that have to be captured from the network elements may be enormous. This huge amount of data may be linked in some way. One advanced pre-processing technique may aim to extract the hidden relationship between the data. To do so, a co-clustering can be applied. Co-clustering aims to identify block patterns in a data set from a simultaneous clustering of rows and columns. Thus, the large data matrix can be summarized by a reduced number of blocks of data (or co-clusters). Let $x = (x_{ij})i \in I;j \in J$, where I is a set of N observations (rows, objects) and J is a set of F attributes (columns, features, variables). The basic idea of co-clustering can be seen as making permutations of objects and variables in order to draw a correspondence structure on IxJ. For illustration, consider Figure 34 that represents a binary dataset of N = 400 observations and of F = 120 binary attributes. By permuting the rows and columns, the dataset is re-organized into a set of 4 x 3 co-clusters, defining 12 blocks of homogeneous data.

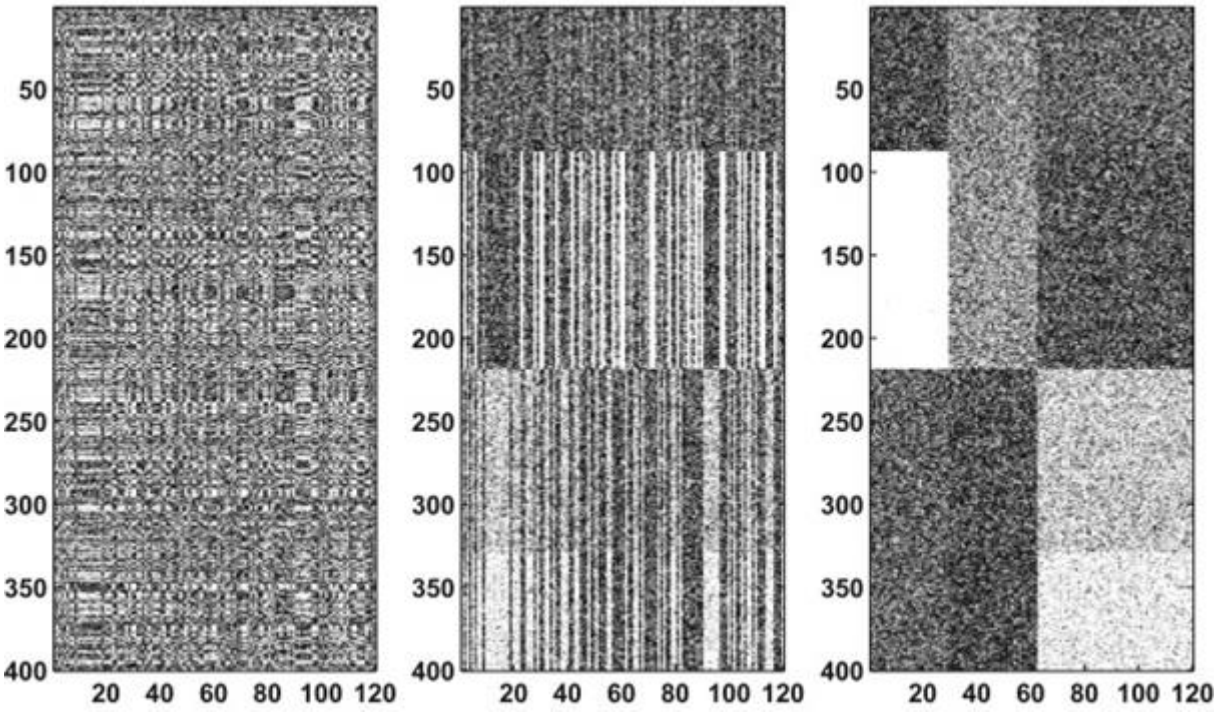

Figure 34 Clustering vs Co-clustering

A co-clustering technique can be applied in order to cluster days of observation, cells and daily evolutions of QoS indicators. Thus, crossing days-clusters and QoS-clusters will lead to define homogeneous blocks of data, containing daily QoS observations having the same behaviour for some group of cells. The obtained block structure may describe the observed data while resuming it. It also helps in defining new macro-QoS indicators.

The data under study are a sample of N observations. Each observation is described by a set of F curves (daily QoS indicators). The statistical model underlying data, represented by multivariate curves, is a stochastic process with continuous time:

$$\mathbf{X} = \{\mathbf{X}(t)\}_{t\in[0,T]} \quad \text{with} \quad \mathbf{X}(t) = (X_1(t),\ldots,X_F(t))' \in \mathbb{R}^F, \quad F \geq 2.$$

Recently a co-clustering technique for functional data has been proposed in [10]. The model is based on the Latent Block Model (LBM, [8]) and it has been adapted for functional data. The LBM assumes local independence, i.e. the N x F curves are assumed to be independent once the row and column partitions are fixed. A smoothing of the raw data with B-spline basis is first applied. Since the notion of probability distribution for functional data is not well defined [8], a Functional Principal Components Analysis (FPCA, [4]) is used in order to plug the functional data into a finite dimensional space. Once each curve is being identified by its principal components, the probability distribution of these latter can be modelled by a multivariate (Gaussian) distribution with block-specific parameters. These latter can be estimated by a stochastic Expectation-maximization algorithm embedding a Gibbs sampling (SEM-Gibbs). For more details, we refer the reader to the paper presented in [10].

The result of the co-clustering approach is blocks of daily evolution of QoS indicators, of days of observations and of cells under study. These co-clusters may allow creating macro-QoS indicators specific to a group of cell. A labelization of each co-cluster is needed and this new information may help in the classification step in order to predict the QoE from these new macro-QoS indicators.

6.4 Machine learning models for anomaly forecasting and noisy neighbor prediction

6.4.1 An "intelligent sensor" for anomaly forecasting

The objective in this example is to allow the QoE sensor to anticipate the anomalies in the RAN part of the network as illustrated in Figure 35, using a supervised learning. With this anomaly prevision, the QoE sensor could have the chance to forecast the malfunctions in the network and to correct the problem before it occurs. Thus, the sensor can save the quality of services and it can guarantee a better quality of experience. The importance of this model for SliceNet project is that it can be applied for Smart Grid, Smart city and eHealth use cases since they aim to offer the best services to their end users, therefore forecasting their service degradation seems inevitable. The data used by the proposed model are QoS indicators and more specifically, daily evolutions of Key Performance Indicators (KPIs) that are considered as functional data.

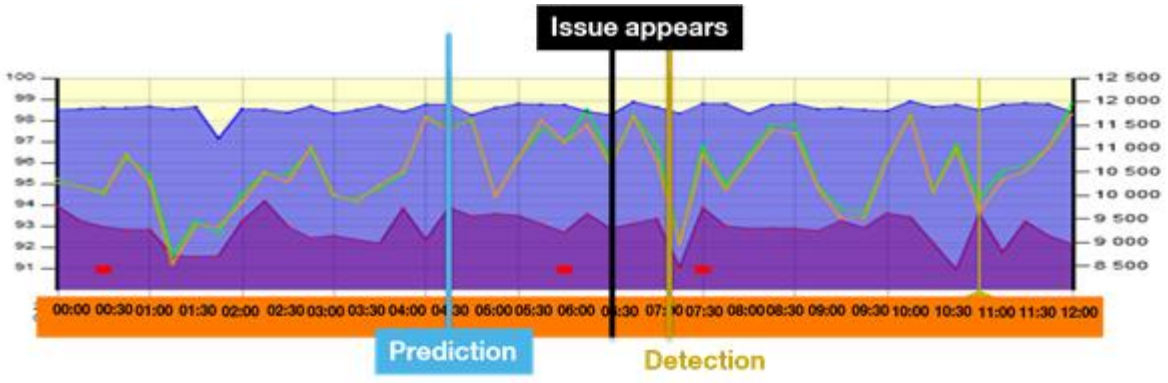

Figure 35 Anomaly forecasting vs anomaly detection

The proposed method offers a supervised technique. It is assumed that a labelled dataset is provided. The dataset is related to a specific use case (the target anomaly to detect) with the corresponding KPIs. The model is window-based: the size of the window, the step of the window as well as the prediction horizon are parameters to be set. Each observation corresponds to the set of the different KPIs curves, for one cell and for one window. The labelling is assured by observing if an anomaly will occur in the prediction horizon. Hence, the data *X* under study are a sample of *n* observations. Each observation X_i is described by a set of *p* curves and a label. The curves are the functional features that correspond to the daily evolution of *p* KPIs as illustrated in Figure 36.

			KPI 1	KPI 2	...	KPI p	Label
i=1	Cell #1	window #1 (day #1)			...		1
i=2		window #2 (day #2)			...		0
		...	...	...	...	...	...
		window #7 (day #7)			...		1
	...	...	...	...	...	...	...
...	cell #c	window #1 (day #1)			...		0
		window #2 (day #2)			...		1
		...	...	...	...	...	...
i=n		window #7 (day #7)			...		0

Figure 36 The structure of the training set

Since the collected KPIs have discrete values, the first step of the proposed model is to retrieve the functional nature of the KPIs by using a smoothing technique. By considering that the numbers of the observed KPIs and cells could be huge and that the observation duration could be long, a dimensionality reduction seems inevitable. Therefore, a Functional Principal Components Analysis should be applied. Once the labelled dataset is resumed in terms of principal components, a classification algorithm can be applied. This classification allows predicting future malfunctions given the observed KPIs for a specific cell and for a fixed window. It can be achieved by any suitable classification algorithm that deals with numeric variables. Examples of the algorithms that can be used are neural networks, support vector machines and decision trees. Figure 37 summarizes the steps for the training phase of the model.

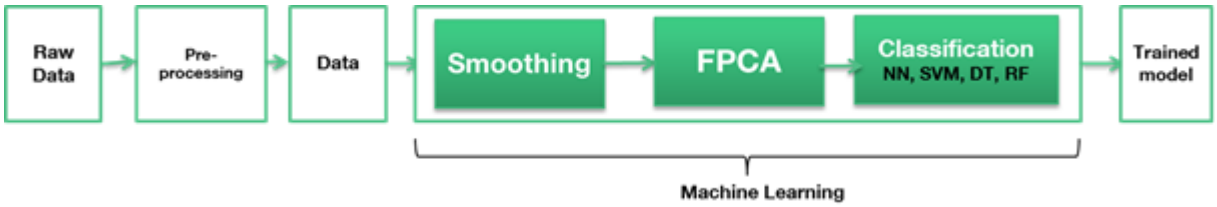

Figure 37 Training phase

Given the smoothing basis, the FPCA basis and the learned classification model, prediction of new observations is possible. For each new observation, a smoothing is first applied for every KPI using the same base as the training phase. The observation in terms of functional features is then projected on the same FPCA basis (used in the training phase). The learned model will then predict if there will be any anomaly in the prediction horizon given the obtained principal components. The prediction phase is illustrated in Figure 38.

Figure 38 Prediction phase

In order to validate the model, three experiments have been conducted in a real LTE network in an urban area with a population of nearly one million. It corresponds to Lyon, a big city in France. The first experiment aims to detect call drop problems. The second experiment aims to detect accessibility anomalies by analysing radio and S1 bearers' setup success rate. The third experiment aims to detect capacity degradation through delay and throughput. The KPIs that we observed for each experiment are described in Table 6.

Table 6 The KPIs used for each experiment

Problem to forecast	Used KPIs
Call drops	UE CTX DROP RATE ERAB DROP RLF RATE
Accessibility	S1 CNX ESTAB SR RRC CNX ESTAB SR
Capacity degradation	CONG DL CAUSE PDCCH RATE UE RRC CONNECTED AVG PERCELL AVG USER MAC UL THROUGHPUT

The data was extracted from 704 cells. The period of observation covered two weeks from December 5, 2016 to December 18, 2016. This period contains some ordinary workdays, weekends, the beginning of holidays and a special event of festival of lights that has been held in Lyon from December 8 to December 10. Therefore, normal behavior of the network can be found in this period as well as problematic observations due to the huge number of users in the cells.

For all the experiments, the size of the window is fixed to one day. The step of the window is equal to the prediction horizon so that neither replications nor holes are possible. The KPIs are extracted with a granularity of 15 minutes (therefore, each daily KPI contains 96 values).

The data contains missing values. They are easily treated by applying a smoothing for each daily evolution of KPIs, since it allows gaining the functional behaviour of the daily KPIs which is an advantage when dealing with functional data. The used smoothing basis is B-splines where the number of basis functions is empirically set to 20. A FPCA is then applied. The number of principal components is chosen so that at least 80% of the information is covered. The horizon prediction varies from 3 hours to one week. 80% of the labelled dataset is used for the training phase and the remaining 20% is used for the test phase. The classification is achieved with a decision tree. The evaluation of the model is held in terms of the performance indicators that are described in Figure 39.

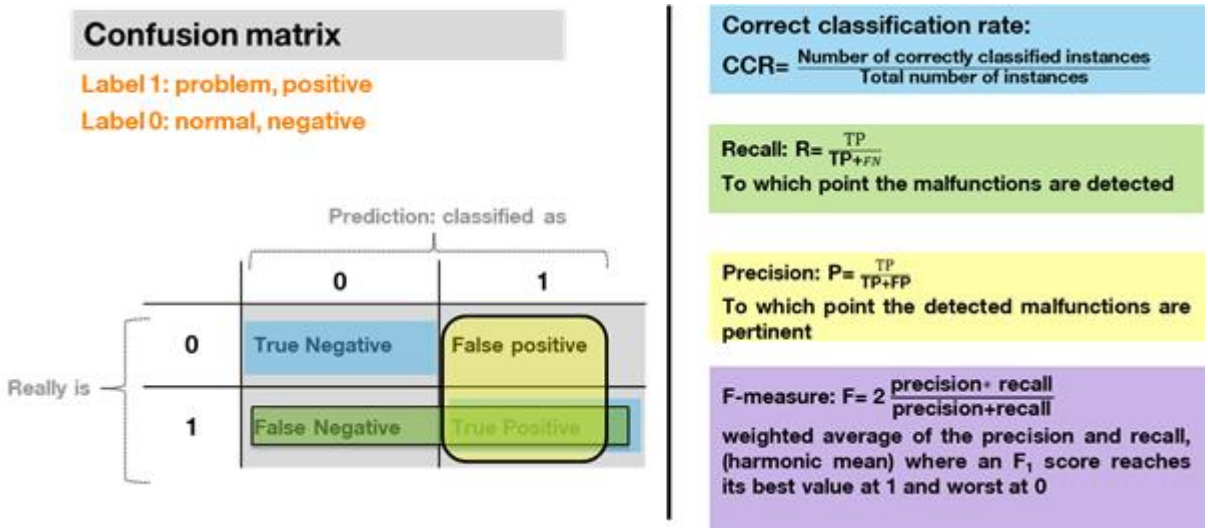

Figure 39 Performance metrics used for evaluating the provision model

The experimental results are illustrated in Figure 40. It presents the correct classification rates and the F-measure results of the prevention model applied to the three experiments.

We notice that the algorithm has the potential to prevent from future problems since the correct classification rate is greater than 70% even for a long prediction horizon equal to one week. The model does not have a tendency to predict a specific class nor to generate false alarms as proven by the F-measures. The performance of the model decreases when the prediction horizon increases which is expected. The data does not allow having a prediction horizon bigger than one week since the extraction covers two weeks. The results are promising since they prove that the model allows forecasting future anomalies with a high precision. It can be integrated in the QoE sensor in order to prevent from performance degradation for the three use cases (Smart grid, Smart city and eHealth).

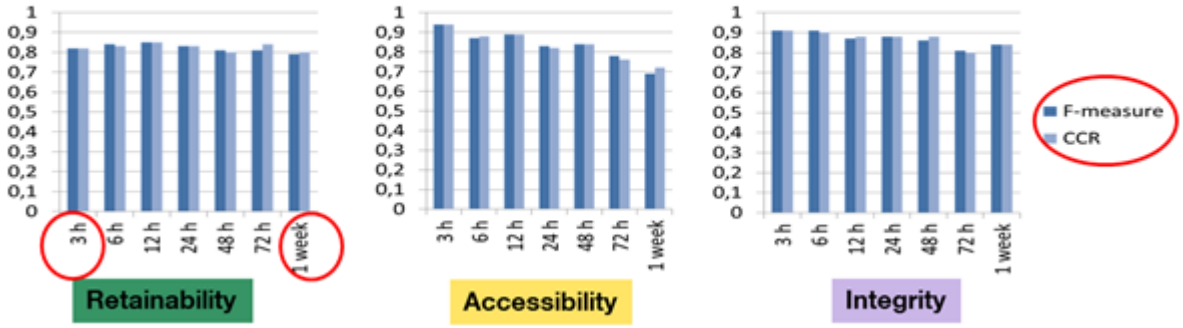

Figure 40 Experimental results for call drop (retain ability), accessibility and capacity degradation (integration)

6.4.2 An "intelligent sensor" for noisy neighbor prediction

5G and network slices will be characterized by billions of heterogeneous and connected devices and rely massively on software networks and virtualization. The network functions composing network slices will be deployed mostly as VNFs in virtualized environments. In the context of SliceNet, we should take into account performance degradation in virtualized environment due to noisy neighbour problem.

The noisy neighbour phenomenon describes the situation in virtualized environment where VNFs or VMs running on the same physical machine compete for resources such as memory, CPU or network bandwidth, resulting in a degradation of performance. The problem of noisy neighbour is considered as the ability for one instance to degrade the performance of other co-located instances.

In this section we will describe steps toward the generation of the model that can be seen as an intelligent sensor that predicts the status of the VNF running in NFV environment.

Our objectives were firstly to create noise in virtualized environment in order to monitor the infrastructure and collect enough data, which can help us to detect and predict the VMs suffering from noise using supervised learning algorithms of machine learning.

To collect data, we used Prometheus monitoring system to gather time-series based numerical data. To monitor a specific VM/VNF, we installed a Prometheus endpoint (node-exporter) over it. This endpoint is a HTTP interface that exposes a list of metrics and the current value of the metrics.

Prometheus server collects the metrics from agents over HTTP, and then stores them. During experimentations, the following metrics were collected:

- CPU utilization
- Usage of memory
- Inbound network traffic
- Outbound network traffic

CPU usage: For CPU request, it has been various modes to monitor, the table below explain each of them:

Table 7 The meaning for each CPU mode

CPU mode	Meaning
user	The time spent in user-land
system	The time spent in the kernel
iowait	Time spent waiting for I/O
idle	Time the CPU had nothing to do
irq & softirq	Time servicing interrupts
guest	If you are running VMs, the CPU they use
steal	If you are a VM, time other VMs "stole" from your CPUs

For our case since we put the node exporter at the level of the virtual machines, instead of the hypervisor, the “steal” and “guest” modes aren’t useful. Thus, to measure the noise affected on the VNF node, we monitored the CPU in “idle” mode. We observed 98% - 100% in CPU utilisation (see Figure 41) which means that the VNF is suffering from noise – the noisy VM negatively impact VNF’s performance – To retrieve data of the CPU usage from the NSP level, data can be aggregated and collected using the following query. This query will be send to the Prometheus server:

$$100 - (\text{avg by}(\text{instance})(\text{irate}(\text{node\_cpu\_seconds\_total}\{\text{mode} = \text{"idle"}\}[5\text{m}])) \times 100)$$

In this expression, we used the Prometheus function node_cpu_seconds_total which tells us how many seconds each CPU spent doing each type of work presented in the table above over 5 min window. Then, to calculate the per-second values we use the irate( ) function. After that, we aggregate to get the overall value across all CPUs for the machine. As these values always sum to one second per second for each CPU, the per-second rates are also the ratios of usage. We subtract the idle usage from 100% to calculate the percentage of CPU used.

Figure 41 The CPU usage during the no-noise (50%) and noise (100%) scenario

Memory utilization: The query that outputs the average memory usage for instances over each 5 min is presented as follow:


```
100 × (1 − ((avg_over_time(node_memory_MemFree_bytes[5m] )
            + avg_over_time(node_memory_Cached_bytes[5m] )
            + avg_over_time(node_memory_Buffers_bytes[5m] ))
            ÷ avg_over_time(node_memory_MemTotal_bytes[5m] )))
```

This query ultimately provides an overall metric for Memory usage. It does this by a calculation based on metric of free, cached, buffers memory as summation, divided by the total memory working out the overall percentage that displays the current memory usage.

Network traffic: For the network traffic, we monitored the inbound and outbound metrics which means respectively as from the perspective of the machine in question; the packets which originate elsewhere and arrive at the machine, and packets which originate at the machine and arrive elsewhere. Here are the queries used:

```
rate(node_network_receive_bytes_total[5m])
```

```
rate(node_network_transmit_bytes_total[5m])
```

The function calculates the per-second average rate of time series in a range vector. With these two expressions, we can get the rate of the total received (inbound) and transmit (outbound) packets on bytes occurred in last 5 min.

After collecting data from Prometheus API, data cleaning is needed, it is the process of standardizing the data to make it ready for analysis. Most of times, there will be discrepancies in the captured data such as incorrect data formats, missing data, errors while capturing the data. This is an important step because the accuracy of the results depends heavily on the data we use. Once data are cleaned, we labelled each instance to one of these following statuses:

1. **Noise**: present the situation where the collocated VMs, existing in the same server, compete for the available resources which lead to a degradation of performance.
2. **No-noise**: present the absence of noisy neighbor phenomenon.
3. **Overload**: define the situation where the VM/VNF is stressed by applications running on it.
4. **Overprovisioning**: describes the situation where a VM is allocated in an over provisioned server (a server that runs out of resources). The overprovisioning issue is another definition of noisy neighbor which have been observed during our experimentations.

The labelling is done according to each scenarios (noise, no-noise, overload and overprovisioning) that we experienced separately. For example, in the overload scenario we stressed our monitored VM/VNF, and then we labelled the collected data supervised during this time slot into ‘overload’ status. The final step is to structure data into arrays in order to apply the machine learning algorithms. The arrays contain data, features names, targets and targets names. Figure 42 shows an overview of all followed steps.

Figure 42 Data preparation steps

After labelling the dataset with the 4 defined targets: noise, no noise, overload and overprovisioning, using pandas library, we structured the data in csv files in order to apply the supervised learning algorithms. For the noisy neighbour use case the Random Forest was used to learn and predict the noisy neighbour phenomenon in NFV environment. This algorithm creates the forest, which is a number of trees, and makes it somehow random. In other words, Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It splits the datasets into small subsets with different size and different branches; each subset is a decision tree that predicts the output. The final predicted class is the majority voting of all subsets. To measure the prediction performance, we used scikit-learn library. The simulation results show that the Random Forest algorithm can detect the noisy neighbor phenomenon with an accuracy of more than 95%.

The model for the noisy neighbour detection was developed in python. It includes data extraction from Prometheus API which is represented in JSON format. For the ML algorithm, it was developed using scikit-learn library. For data cleaning and labelling we have used pandas and glob libraries' python. The code of the noisy neighbor model will be published on github or gitlab when WP8 platform is available.

7 Conclusion and link with SliceNet WPs

The assumption of deliverable 5.2 was to design QoE Sensor based on the work conducted in the general framework of QoE management (T5.1) and in vertical definition (T2.1).

Specific QoE models for the individual use cases should be optimised based on the reference model defined in T5.1. One of challenges for T5.2 was the lack of QoS data related to use-cases (smart grid, smart city and eHealth) and workflows definitions are not yet finalised. Therefore, it was agreed to focus on the design of the QoE Sensor in this deliverable.

Hence to ensure the agility, coherency and the fluidity of the results, D5.2 focuses on the design including the machine learning approach for intelligent QoE sensors.

The D5.2 as the first deliverable to be submitted from the WP5 activities will be the core stones input to D5.3, D5.4 and the global D5.1. In fact, the implementation will take place following the design defined in D5.2 and pushing further the application of the machine learning techniques, where appropriate, to the project use cases when the data is collected. As following, the QoE sensors implementation will be in the other WP5 deliverables, their integration will take place in WP8 per use-case and WP6 will provide QoS data collected at the NSP level.

8 References

[1] G. E. P. Box, G. C. Reinsel and G. M. Jenkins. Time series analysis: forecasting and control. Prentice-Hall, Englewood Clis, NJ, 1994.

[2] A. Beygelzimer, E. Erdogan, S. Ma and I. Rish. Statistical Models for Unequally Spaced Time Series. 04 2005.

[3] M.Schulz and M. Mudelsee. REDFIT: estimating red-noise spectra directly from unevenly spaced paleoclimatic time series. Computers Geosciences, vol. 28, no. 3, pages 421-426, 2002.

[4] J. O. Ramsay and B. W. Silverman. Functional data analysis. Springer Series in Statistics. Springer, New York, second edition, 2005.

[5] F. Ferraty and P. Vieu. Nonparametric functional data analysis. Springer Series in Statistics. Springer, New York, 2006.

[6] J. O. Ramsay and C. J. Dalzell. Some Tools for Functional Data Analysis. Journal of the Royal Statistical Society. Series B (Methodological), vol. 53, no. 3, pages 539-572, 1991.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Int. Res., vol. 16, no. 1, pages 321-357, June 2002.

[8] A. Delaigle and P. Hall. Defining probability density for a distribution of random functions. The Annals of Statistics, vol. 38, pages 1171-1193,2010.

[9] G. Govaert and M. Nadif. Latent Block Model for Contingency Table. Communications in Statistics-Theory and Methods, vol. 39, no. 3, pages 416 - 425, January 2010.

[10] Y. Ben Slimen, S. Allio and J. Jacques. Model-based co-clustering for functional data. Neurocomputing, vol. 291, pages 97 - 108, 2018.

[11] P. Reichl, S. Egger, R. Schatz, and A. D'Aiconzo, "The logarithmic nature of QoE and the role of the Weber-Fechner law in QoE assessment," 2010, pp. 1-5.

[12] P. C. Ramosl, 1. N. Salmernl, R. P. Leal, and F. G. Vidal, "Estimating perceived video quality from objective parameters in video over IP services," in The Seventh International Conference on Digital Telecommunications. ICDT, 2012, pp. 65-68.

[13] G.I. Crabb, R. Beaumont, D. Webster, “Review of the class and quality of street lighting”, CSS Street Lighting Project Report, 2009.

[14] P. Brooks and B. Hestnes, “User Measures of Quality of Experience: Why Being Objective and Quantitative is Important,” IEEE Network Magazine, vol. 24, no. 2, pp. 8-13, March 2010.

[15] ITU –T E.800 Definitions of terms related to quality of service, 2009

[16] ITU –T P.10/G.100 Vocabulary for performance and quality of service, 2017

[17] Qualinet White Paper on Definitions of Quality of Experience

[18] SliceNet D2.4 Management Plane System Definition, APIs and Interfaces

[19] SliceNet D2.1 Vertical Sector Requirements Analysis and Use Case Definition

[20] ITU-T G.1010 End-user multimedia QoS categories, 2001

[21] SliceNet D3.4 Design and Prototyping of Integrated Multi-domain SliceNet Architecture

[22] 3GPP TS 23.502 Procedures for the 5G System, 2018

