H2020 5Growth Project
Grant No. 856709

D4.3: 5G facility validation and verification report

Abstract

This deliverable constitutes the second of a series of three, from D4.2 to D4.4, intended to provide the results of the project, addressing not only the verification of pilot outcomes in the context of the successive releases of the experimental environments, but also the establishment of a systematic way to perform this verification, guaranteeing verification quality and contributing these findings and tools for their application elsewhere.

Document properties

Document number	D4.3
Document title	5G facility validation and verification report
Document responsible	Álvaro Gomes (ALB)
Document editor	Álvaro Gomes (ALB)
Editorial team	Carlos Marques (ALB), Carlos Guimarães (UC3M), Giada Landi (NXW), Juan Brenes (NXW), Hugo Martins (EFACEC_E), Aldo Trindade (EFACEC_E), Paulo Paixão (EFACEC_S), Rui Manuel Antunes (EFACEC_S), Pedro Elísio (EFACEC_S), Nikos Maroulis (NKUA), Diego Lopez (TID), Antonio Pastor (TID), Ramón Perez (TELCA), Matteo Pergolesi (TELCA), Olivier Tilmans (NBL), Josep Xavier Salvat (NEC), Ignacio Domínguez (UPM), Daniel González (UPM), Luis Bellido (UPM), David Fernández (UPM), Encarna Pastor (UPM), Giulio Bottari (TEI), Fabio Ubaldi (TEI), Stefano Stracca (TEI), Paola Iovanna (TEI), Andrea Sgambelluri (SSSA), Luca Valcarenghi (SSSA), Daniel Corujo (ITAV), Vitor Cunha (ITAV), João Fonseca (ITAV), João Alegria (ITAV), Diogo Gomes (ITAV), Manuel Lorenzo (ERC), Diego San Cristóbal (ERC), Fernando Beltrán (ERC), Engin Zeydan (CTTC), Jorge Baranda (CTTC), Josep Mangues (CTTC), Ricardo Martinez (CTTC), Luca Vettori (CTTC), Marco Ajmone Marsan (POLITO), Claudio Casetti (POLITO), Carla Fabiana Chiasserini (POLITO), Corrado Puligheddu (POLITO), Jorge Martin (UC3M)
Target dissemination level	PU
Status of the document	Final
Version	1.0
Delivery date	June 30, 2021
Actual delivery date	June 30, 2021

Production properties

Reviewers	Carlos Guimarães (UC3M), Ricardo Martinez (CTTC)

Disclaimer

This document has been produced in the context of the 5Growth Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-856709.

Contents

List of Figures

List of Tables

List of Acronyms

5Gr-SO – 5Growth Service Orchestrator

5Gr-VoMS – 5Growth Vertical-oriented Monitoring System

AGV – Automated Guided Vehicle

AI/ML – Artificial Intelligence / Machine Learning

API – Application Programming Interface

AR – Augmented Reality

CKPI – Core 5G KPI

CMM – Coordinate-Measuring Machine

CP – Control Plane

CPE – Customer Premises Equipment

CUPS – Control and User Plane Separation

DB – Database

DSS – Decision Support System

E2E – End-to-End

ELK – Elasticsearch

eMBB – enhanced Mobile Broadband

FR – Functional Requirements

gNMI – gRPC Network Management Interface

I4.0 – Industry 4.0

IIoT – Industrial Internet of Things

KPI – Key Performance Indicator

LL – Logical Link

LV – Low-Voltage

LX – Level Crossing

mMTC – massive Machine Type Communications

MTTR – Mean Time To Repair

NBI – Northbound Interface

NS – Network Service

NSD – Network Service Descriptor

NSIR – Network Service Instance Resource

PA – Placement Algorithm

RAN – Radio Access Network

RL – Resource Layer

ROE – Resource Orchestration Engine

SDA – Semantic Data Aggregator

SKPI – Service KPI

SLA – Service Level Agreement

SO – Service Orchestrator

SOE – Service Orchestrator Engine

TMV-WG – Test, Measurement and KPI Validation Working Group

UC – Use Case

UE – User Equipment

UP – User Plane

URLLC – Ultra-Reliable Low Latency Communication

vEPC – virtual Evolved Packet Core

VIM – Virtualized Infrastructure Management

VM – Virtual Machine

VNF – Virtual Network Function

VPN – Virtual Private Network

VS – Vertical Slicer

ZDM – Zero Defect Manufacturing

Executive Summary and Key Contributions

This deliverable constitutes the second of a series of three, from D4.2 to D4.4, intended to verify pilot results, aligned with the successive releases of the experimental environments developed in WP3 and their integration of the innovations addressed by WP2. These pilot results are verified in a systematic way, supported by a methodology and a set of tools reported in this series of deliverables as well.

The main contributions of this deliverable can be listed as:

1. Definition of the data infrastructure, introduced in D4.2, and responsible for enabling an experimental data processing framework that collects measurements from the monitored network infrastructures, aggregates the monitoring data, and eventually delivers the aggregated data for further analysis.
2. Identification of the 5Growth experiment catalogue requirements and presentation of selected entries from the 5Growth experiment catalogue stored in the project source repository. The 5Growth experiment catalogue aims to enable the reproducibility of experiment results.
3. Description of the Information Models (IMs) of the 5Growth Experiment Descriptors, which are based on an evolution of the Experiment Blueprints defined in the 5G EVE project.
4. Identification and characterization of the technical requirements for the different pilots, the KPIs that are measured in the pilot validation campaigns, and the measurement procedures to execute them.
5. Description of the integration process of the 5Growth Monitoring platform with the data flows produced by experiments in the context of both 5G EVE and 5G VINNI facilities.
6. Identification of the project innovations that are integrated on each pilot and use case.
7. Report the results of the second validation campaign performed on each pilot, including faced constraints, used tools, measurement results and analysis of the achieved results.

D4.3 follows the structure defined for D4.2 and that will be used for the last WP4 deliverable (D4.4). It is arranged around the same idea of combining reports on the evolution of methodology and tooling, and the analyses of the verification results. WP4 foresees these contents will evolve during the project lifetime, dedicating more space and detail to result verification as both the methodology matures, and the experimental capacity of the different pilots consolidates.

1. Introduction

WP4 main objective is to evaluate core (network) and service (application) KPIs through the 5Growth field trials, validating the applicability of 5G technologies to the different use cases considered by the project pilots. These does not only include the execution of the different verification campaigns, but also the definition of the measurement and testing methodologies applied in the campaigns. WP4 is committed to provide a detailed description of these methodologies and a series of tools to enable the automation of the verification procedures, the processing of the measurement data and the reproducibility of the verification results. This way, WP4 is focused not only on performing pilot verification, but also on doing so in a systematic way. The latter would guarantee scientific (engineering) quality and contribute to these findings and tools for their application elsewhere.

The work in WP4 is aligned with WP2 and WP3 activities, addressing three innovate/deploy/validate cycles during the project lifetime. WP4 applies the appropriate releases produced by WP3, that incorporate the modules produced in WP2, and selects the applicable experimental environment for the corresponding cycle, comprising:

1. Data sources and consumers, including the metadata for them.
2. External tools to be applied, available through the project tool catalogue.
3. Specific measurement methods, documented in the deliverables.
4. Pilot use cases are validated in the context of the successive releases of the experimental environments, and reported in the associated deliverable, including the environment characteristics and the outcome of the verifications run during the period.
5. The verification methodology and associated tooling provide the common substrate to the execution of verification campaigns, working to:
6. Enhance the quality of verification campaigns.
7. Guarantee repeatability and reproducibility as essential features of experimental reports.
8. Contribute to a better understanding of the mechanisms to aggregate and process telemetry data for data-enabled network management.

The validation campaigns are intended to collect evidence about the fulfilment of the core 5G and service KPIs in the contexts of the different pilots, analysing the collected data and producing reports to verify the applicability of 5G technologies in the associated scenarios.

Therefore, this deliverable is structured as follows:

A section describing the data infrastructure responsible for enabling an experimental data processing framework that collects measurements from the monitored network infrastructures, the 5Growth experiment catalogue including the Information Models (IMs) for the 5Growth experiment descriptors, and the new tools added to the toolset supporting the pilot validation campaigns.

A section identifying pilot technical requirements and the KPIs measured in the pilot validation campaigns, together with the innovations integrated in each pilot or use case. The section describes the integration process for making the monitoring data, collected by experiments in the context of 5G EVE and 5G VINNI, available to the 5Growth Monitoring platform.

A section for each pilot validation campaign, addressing the description of the setup of each use case, measurements and analysis of results taking into consideration, as reference, the specific technical and performance requirements. The implementation of the pilots is an ongoing process characterized by successive incorporation of new features and functionalities targeting better performance and services to verticals. The evaluation is conducted into in three moments of the project lifetime, corresponding to the three project validation campaigns. The evaluation results and conclusions for the second validation campaigns are one of the most important contents of this document.

2. Measurement Tools and Methodology

This section presents the progress in the methodologies and tools applied for performing measurements and in processing, integrating, and making available the produced data for further applications. The evolution of the data infrastructure, that D4.2 provided an initial design for, is introduced, describing the design choices and available modules for modeling data sources and consumers, and the mechanisms for forwarding the data, once pre-processed, according to the applicable metadata descriptions. Further on, an update on the measurement tools developed and/or applied within the project is provided, as an incremental report with respect to what was described in D4.2. The section also includes the definition of experiments descriptors, a formal model for experiment definition intended to guarantee reproducibility (across different laboratory environments) and repeatability (under controlled variations of the experimental conditions) of the 5Growth experimental results. The integration of these descriptors within an experiment catalogue (able to support formal referencing of experimental results) completes the section.

2.1. Data Infrastructure

The data infrastructure enables a monitoring framework collecting measurements from the monitored network infrastructures, aggregating the monitoring data, and eventually delivering such data for further analysis. To this end, the data infrastructure is built around a main functional block called Semantic Data Aggregator (SDA).

2.1.1. Semantic Data Aggregator (SDA)

The SDA is a data engineering platform that facilitates the exchange of data among data sources and data consumers. In this process, the platform transforms the data models used by the data sources into the data models expected by the data consumers. Thus, data sources and data consumers do not need to implement one-to-one adaptions for every use case. As a result, the SDA provides a scalable monitoring platform to which data sources and data consumers can be dynamically plugged.

In addition, the SDA is responsible for recording the flow of the data as it moves from the data sources to the data consumers, i.e., data lineage. The SDA describes how data has been produced, transformed, and eventually consumed, by leveraging ETSI ISG CIM (Context Information Management) standards [1] as the means for managing the metadata that characterizes each of the steps in the flow.

The Figure 1 depicts the architecture of the SDA platform:

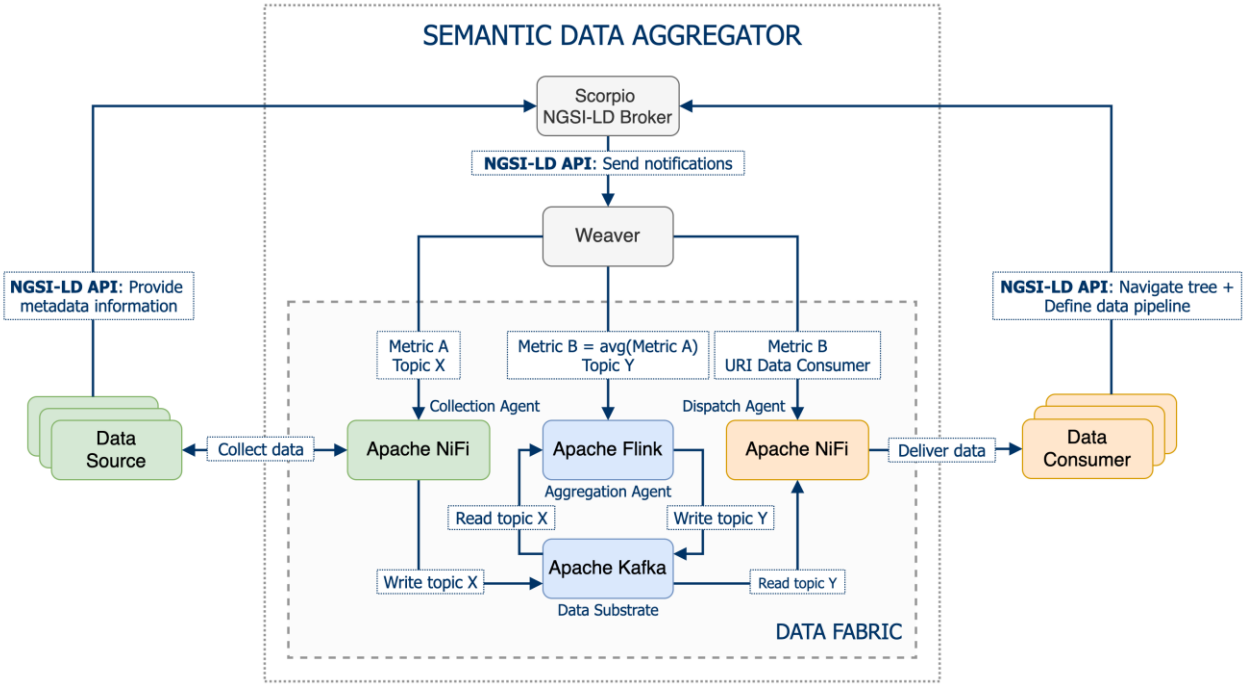

FIGURE 1: ARCHITECTURE OF THE SEMANTIC AGGREGATOR

The SDA is composed of three main building blocks: the Data Fabric, the Weaver, and the Scorpio NGSI-LD Broker.

Data Fabric

The Data Fabric is responsible for integrating data sources and data consumers that wish to exchange data through the SDA. The Data Fabric implements the required mechanisms to collect, aggregate, and dispatch data by means of the so-called agents. Different data engineering projects have been chosen depending on the nature of the agents. The following list briefly introduces each of these projects and how they fulfil the role of each agent:

- **Apache NiFi**
  The purpose of the collection and dispatch agents is two-fold: first, implementing the protocol for transporting the data; second, parsing the data as per the prescribed data model. In this regard, Extract-Transform-Load (ETL) tools fit perfectly. These tools focus on retrieving data from a source, applying transformations such as data cleansing or data wrangling, and finally, storing the resulting data into a sink storage. The Data Fabric leverages Apache NiFi [2] as an ETL solution that can play the role of the collection and dispatch agents.

  Apache NiFi is a popular ETL system that facilitates a secure and reliable distribution of data. Apache NiFi supports programming of data flows as graphs. NiFi provides out-of-the-box more than 200 processors for multiple purposes namely data collection, data delivery, schema transformations, data routing, or data enrichment. NiFi also comes with a friendly GUI, and a fine-grained control of data provenance that eases debugging data flows.

- **Apache Flink**
  The SDA aims at enabling the estimation of network infrastructure KPIs in real time. In this respect, stream processing engines such as Apache Spark [3], Apache Flink [4], or Kafka Streams [5], have risen as the most adopted solutions.

  In the case of the Data Fabric, Apache Flink has been selected due to its support for stateful computations over streams of data, and its integration with multiple source and sink substrates such as Apache Kafka or HDFS. In the proposed architecture, Apache Flink plays the role of an aggregation agent by sitting between the collection and the dispatch stages. This agent is responsible for performing typical data transformation operations such as filtering or windowed aggregations.

- **Apache Kafka**
  The exchange of data among the agents of the SDA is achieved by leveraging a message queueing system. In this sense, Apache Kafka [6] has become the de-facto standard for exchanging messages in stream-oriented architectures. Apache Kafka provides a reliable distributed storage with pub/sub capabilities. By leveraging Apache Kafka, the agents of the Data Fabric can dynamically subscribe to specific topics to receive data produced by other agents.

  Figure 1 shows an example where the collection agent writes data into topic X which the aggregation agent is subscribed to. The resulting data produced by the aggregation agent is then written to another topic Y, which the dispatch agent is subscribed to. But this is just a possible combination, as the dispatch agent could also subscribe to topic X if no data aggregation is required for other use cases. Therefore, Apache Kafka enables a flexible architecture that allows different combinations of agents as needed in the data pipeline.

Weaver

The Weaver component is an in-house built component to orchestrate the configuration and instantiation of the aforementioned agents running within the Data Fabric. The Weaver is a Python application based on the FastAPI framework [7] that consumes metadata information in order to configure the agents. As a result, the Weaver facilitates the automatic creation of data pipelines between sources and consumers. For instance, the Weaver orchestrates the instantiation of ETL flows in Apache NiFi, or the execution of stream processing applications in Apache Flink.

In the process of building the data pipeline, the Weaver is also responsible for producing additional metadata based on the progress of the execution of the agents. Thus, enriching the data lineage that describes a pipeline within Data Fabric.

Scorpio NGSI-LD Broker

In order to gather data lineage information of the data flows between sources and consumers, the SDA leverages the ETSI CIM standard for the metadata management. The ETSI CIM standards specify the NGSI-LD protocol, which sources and consumers can leverage to exchange metadata information through a central component called the context broker.

Scorpio [8] is a NGSI-LD compliant implementation of the context broker. Scorpio, which is under development by NEC, has been chosen as the context broker to be used within the SDA due to its support for distributed and federated architectures.

In the example shown in Figure 1: , the data source provides metadata information that characterize itself e.g., endpoint URL. On the other hand, data consumer interacts with Scorpio in order to find and request a subscription to the interested data source. More precisely, the consumer specifies a data pipeline for consuming a particular Metric B which happens to be the time-windowed average transformation of another Metric A. All this information is represented as metadata by means of NGSI-LD protocol, and stored by the Scorpio broker. The Scorpio broker then notifies the Weaver component about the new data pipeline requested by the data consumer. At this point, the Weaver reads the metadata information and triggers the configuration of the Data Fabric agents involved in the pipeline.

2.1.2. Data Sources

The purpose of the SDA is the integration of heterogenous data sources and data consumers and coordinate the exchange and aggregation of the data between them. This section analyzes the data source integration process that enables SDA collecting and aggregating information from these sources.

As a contribution to the project, the data aggregator is being used as a framework that allows the adaptation of different data sources whose information is used for the validation of the 5Growth KPIs. According to deliverable D4.2, these data sources are classified into three main categories:

- **5Growth integrated data sources:** Include all the data sources that have been already fully integrated with the 5Growth Vertical-oriented Monitoring System (5Gr-VoMS). The SDA will be responsible for associating external data sources to the 5Gr-VoMS. The SDA will be also able to collect and add either metrics or monitoring data ingested from the 5Gr-VoMS.
- **ICT-17 data sources:** Include the data sources provided by the monitoring systems available in the ICT-17 platforms the 5Growth architecture interacts with (i.e., 5G-EVE and 5G-VINNI). Their integration with the 5Gr-VoMS requires the implementation of specific data adaptation blocks. These blocks perform the translation between the information/data models and the monitoring data collection procedures adopted in the 5G-EVE and 5G-VINNI platforms and the ones at the 5Gr-VoMS. Precisely, the SDA constitutes this adaptation building block in charge of integrating these data sources.
- **Other external data sources:** Embrace data sources generated from any other element external to the 5Growth environment and its related ICT-17 sites. As for the previous case, the SDA will be responsible for the adaptation and processing of this kind of monitoring data to enable their usage on the 5Gr-VoMS.

According to these categories, three specific input data sources were identified for the adaptation with the semantic data aggregator:

- **Prometheus-based data sources:** Prometheus [9] is identified as an important data source within the first category, as the 5Gr-VoMS is based on a Prometheus server.
- **5G EVE Kafka broker data source:** Within the ICT-17 projects, for the integration of both the 5Growth and 5G EVE monitoring platforms, the Kafka broker of the Data Collection Manager (DCM) component corresponding to the 5G EVE platform has been identified as the input data source from where the necessary monitoring data to be consumed by the 5Growth platform is collected. For this, the SDA subscribes to the Kafka's broker topics of the 5G EVE Monitoring Platform. Specifically, this integration between 5G EVE and 5Growth monitoring platforms is necessary for the INNOVALIA pilot experiments.
- **Telemetry-based data sources:** For the external data sources category, YANG-based network devices have been defined as examples of telemetry data sources.

The integration of the data sources with the semantic data aggregator requires the configuration of collection agents. These agents need to address the following two aspects:

1. **Transport:** Mechanism that collects data from the source. This mechanism could be either push-based or pull-based operation/function.
2. **Encoding:** Data is sent from the source in a particular format which may follow a structure. The agent must parse and transform the data into a format that is suitable for future consumption.

Regarding the encoding aspect, the collection agent leverages a special component called the driver which transforms the ingested events into a YANG data model. By "translating" the event into a YANG data model tree, the SDA structures the data using the encoding format of best interest such as Apache Avro [10] or Protobuf [11]. Notice here the use of the term event. Each YANG data model is specifically built to cope with individual events collected from a data source. Hence, the role of the driver is the transformation of data on an event basis. Beware that this approach requires the collection agent to include additional mechanisms to pre-process the ingested data to provide the driver with individual events.

Once the driver finishes the job, the collection agent applies the required mechanisms that write the encoded structured data in the Data Substrate (e.g., the SDA Kafka broker).

In order to facilitate the adoption of new data sources, a methodology for implementing the so-called data source drivers is proposed. Figure 2 depicts a first approach to this implementation.

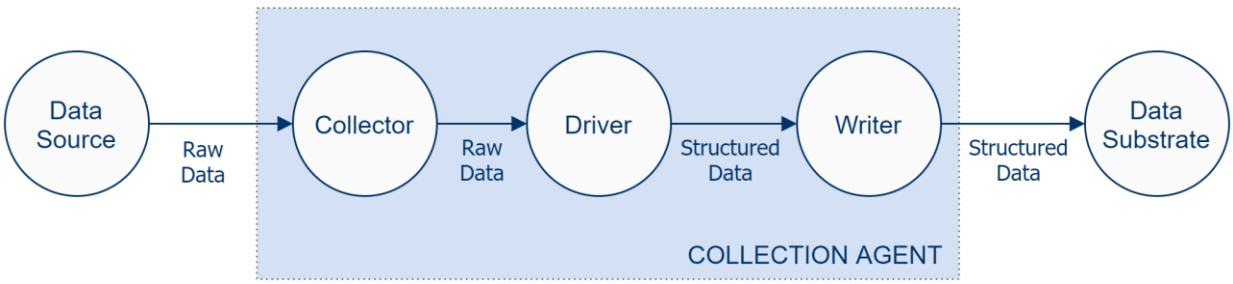

FIGURE 2: DATA SOURCE DRIVER

This approach is agnostic to the chosen tools. In the case of the current semantic data aggregator implementation, Apache NiFi, Apache Kafka, and Apache Avro are the chosen pieces for building the data source drivers.

The following subsections will analyze in detail the data collection process carried out by the semantic data aggregator for each of the indicated data sources.

Prometheus-based data sources integration

Prometheus server is one of the main components of the 5Gr-VoMS to collect metrics from the monitored targets. Indeed, Prometheus has become one of the most relevant monitoring systems for metric collection. Therefore, Prometheus can be considered as one of the most widespread data sources to be considered in the project.

For the Prometheus-based data source use case, the data collection process basically consists of subscribing to specific metrics at a particular polling interval from a Prometheus server whose endpoint service is accessible via URL.

The SDA needs to configure a collection agent that allows the integration with the Prometheus data source. This collection agent works as a NiFi processing flow. Therefore, the NiFi flow configuration describes the collection process from Prometheus metrics. The steps for the collection process are as follows:

1. First of all, the NiFi flow starts with an *InvokeHTTP* processor that allows interacting with a configurable HTTP endpoint. This endpoint is the Prometheus service REST API. The *InvokeHTTP* processor polls the Prometheus API for specific metrics and returns the JSON data model representation for each requested metric that will be consumed by the following processors in the NiFi flow.
2. The following NiFi processors in the processing flow are responsible for cleaning and transforming the representation of the Prometheus metrics. In particular, it seems necessary highlighting the importance of separating the metrics into different time series, where each time series will involve a single event in the Data Substrate (i.e., the SDA Kafka broker). These processors also perform the essential function of transforming the Prometheus metric model into a standard JSON representation according to the corresponding Avro serialization schema (closely related to the process of implementing drivers for data sources detailed above).
3. Finally, the NiFi processing flow serializes the Prometheus metric time series samples to an Avro binary format according to the schema. After the serialization is done, another NiFi processor is responsible for publishing the samples as new events within the specific Kafka topic associated with the Prometheus metrics collection process.

5G-EVE Kafka broker data source integration

An interesting use case for the interaction between the monitoring systems available on both the ICT-17 platforms and the 5Growth platform is to support the 5G EVE and 5Growth monitoring platforms interconnection. To achieve this interconnection, the usage of the semantic data aggregator is proposed.

It is interesting to mention that the SDA should allow the translation between the data model followed in 5G-EVE to publish data on its monitoring platform and the 5Gr-VoMS data model (based on Prometheus).

Within the 5G-EVE Monitoring Platform architecture, the DCM is the component responsible for the collection, persistence, and delivery of all the network and vertical performance metrics. The DCM component is based on Apache Kafka and Apache Zookeeper. Basically, the monitoring data is published in a Kafka broker in the form of a list of records, according to the information model followed in 5G-EVE, which is based on a JSON string format.

To support the data collection process, the semantic data aggregator collects the monitoring data by subscribing to the specific Kafka topics within the DCM, and then use the 5G-EVE prescribed data model to process the data. Therefore, the Kafka broker is considered as another important data source within the semantic data aggregator.

As for the previous Prometheus-based data source case, the SDA needs to configure a collection agent that allows the integration with the Kafka broker data source. This collection agent works as a NiFi processing flow. In this integration use case, the NiFi flow configuration describes the collection process from Kafka topic data records based on the specific 5G-EVE Monitoring Platform information model. The steps for the collection process are as follows:

1. First of all, the NiFi flow starts with an *ConsumeKafka* processor that allows consuming messages from Kafka brokers from specific topics.
2. The following NiFi processors in the processing flow are responsible for cleaning and transforming the representation of the collected Kafka messages. In particular, one processor splits the records of the Kafka topic messages, where each record will involve a single event in the Data Substrate (i.e., the SDA Kafka broker). Another NiFi processor performs the essential function of transforming the Kafka record model to a standard JSON representation according to the Avro serialization schema (closely related to the process of implementing drivers for data sources).
3. Finally, the NiFi processing flow serializes the Kafka message records to an Avro binary format according to the schema. After the serialization, another NiFi processor is responsible for publishing each record as a new event within the specific SDA data substrate topic associated with the Kafka data source monitoring record collection process.

Telemetry-based data sources integration

An interesting external data sources in the network infrastructure domain are telemetry-based devices. Model-based streaming telemetry as a monitoring mechanism for network devices is gaining attention, mainly relaying on YANG data models and management protocols like gNMI (gRPC Network Management Interface) [12].

To carry out a proof-of-concept for telemetry-based data source integration, a containerized version of Arista manufacturer's routers called cEOS [13] was chosen. These Arista cEOS routers support YANG-model based configuration management and streaming telemetry over gNMI/gRPC, NETCONF, and RESTCONF transport protocols. For this use case, gNMI has been chosen as the

telemetry-based management protocol. To collect the telemetry information of Arista cEOS routers from the gNMI protocol, the gNMI CLI client called gNMIc [14] has been used. This gNMI client has full support for gNMI RPC operations: capabilities, get, set, and subscribe.

For the telemetry-based data source use case, the data collection process basically consists of subscribing to a specific *XPath* at a particular sample interval from a telemetry-based device whose endpoint service is accessible via URL. The *XPath* is the selector for the specific YANG data node(s) from the YANG model(s) supported by the target network device.

As for the previous data source integration cases, the SDA needs to configure a collection agent that allows the integration with the telemetry-based data source. This collection agent works as a NiFi processing flow. Therefore, the NiFi flow configuration describes the collection process from network devices information based on YANG models. The steps for the collection process are as follows:

1. First of all, the NiFi flow starts with an ExecutionProcess processor that allows executing a subscription to a specific YANG data node or data nodes group from the gNMIc client. The response is generated in a protojson format (based on Protocol Buffer message encoding as a JSON format followed by gNMI protocol). It is important to note that gNMIc returns separate information about each leaf data node from the subscribed YANG tree data node, where each individual YANG data node will involve a single event in the Data Substrate (i.e., the SDA Kafka broker).
2. The following NiFi processors in the processing flow are responsible for cleaning and transforming the representation of the YANG data node. In particular, it is important to highlight the transforming of the YANG data node protojson format representation to a standard JSON representation according to the Avro serialization schema (closely related to the process of implementing drivers for data sources).
3. Finally, the NiFi processing flow is responsible for serializing the YANG data nodes to an Avro binary format according to the schema. After the serialization, a NiFi processor publishes the sample as a new event within the specific Kafka topic associated with the own telemetry data collection process.

2.1.3. Data Consumers

In this subsection the data consumers currently supported by the SDA are introduced. The integrations of the following data consumers are described as generic as possible, leaving the actual implementation details to the different use cases that may derive from the 5Growth pilots.

Anomaly Detection Module (AD)

The Anomaly Detection module (AD) is bound to the innovation 9 for the WP2 [15]. The purpose of this module is the analysis of the RAN measurement KPIs to identify and predict anomalies, such as latency degradation or packet loss, hence enabling a fast recovery within the network.

In order to provide real time analysis of the network, the AD consumes streams of events containing RAN measurements for each monitored cell. The AD module offers a REST-based service for consuming the stream of events. This service is still under development and will support the ingestion

of events encoded with JSON structured as per a specific data model. As of this writing, there is not a final JSON schema, thus we will only list the relevant fields of the data model for the reader's convenience.

TABLE 1: DATA MODEL FIELDS FOR ANOMALY DETECTION MODULE

Field Name	Data Type	Description
ul_delay	Float	Uplink delay (ms)
dl_delay	Float	Downlink delay (ms)
lost_packets	Integer	# of lost packets, per service per user
rsrp	Float	RSRP (dB)
transfer_protocol	Boolean	TCP or UDP encoded [0,1]
urlx_cell	Integer	UE Bytes received from the cell
timestamp	Datetime	Measurement timestamp (datetime format - "yyyy'-'MM'-'dd' 'HH':'mm':'ss")
cell_id	Integer	RAN cell ID

The fields in Table 1 refer to the RAN KPIs required by the AD module for the network analysis. In this regard, the SDA provides the AD with measurements for each of these KPIs. Figure 3 depicts the workflow that takes place:

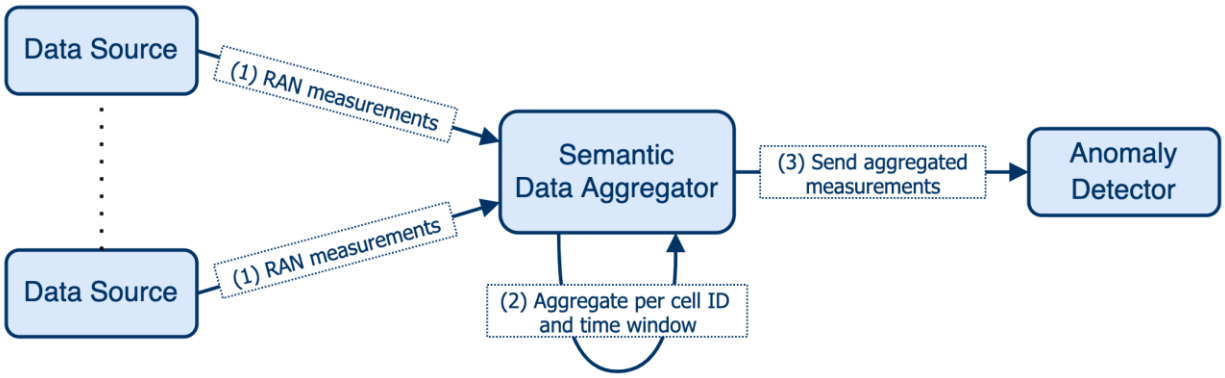

FIGURE 3: AGGREGATION OF RAN FOR THE AD MODULE

In the workflow, the SDA first collects the RAN measurements from those data sources that provide the target KPIs. Then, these measurements are aggregated by RAN cell and by a specific time window. Finally, the SDA bundles the measurements into an event that is first encoded as per the prescribed data model, and then delivered to the REST service of AD module.

The integration of the AD module as a data consumer perfectly illustrates the benefits of the SDA. There could be complex scenarios in which data sources of different natures may monitor only part of the target KPIs. In this sense, the SDA is able to selectively collect RAN measurements from those data sources that provide the KPIs required by the AD module. This is achieved by first, having notion of which data source provides which KPI by means of metadata characterization; and second, aggregating the collected RAN measurements by cell and time windows.

Prometheus

Within the scope of the data infrastructure, Prometheus is considered as a type of data consumer. The integration of the Prometheus data consumer with the SDA can be achieved in two possible ways: Prometheus exporter and Prometheus Pushgateway.

- **Prometheus exporter**

  The exporter is the most recommended method for collecting metrics in Prometheus. In this mechanism, the exporter exposes the available metrics through an HTTP endpoint, and a Prometheus job is configured to periodically scrape the endpoint. An advantage of this method is that multiple Prometheus servers can scrape metrics from the same exporter.

  In the SDA, the Prometheus exporter mechanism must be implemented by the dispatch agent, i.e., Apache NiFi. In this sense, a NiFi flow would be responsible for collecting the generated metrics from Apache Kafka, encode the data into Prometheus's text-based format, and expose the metrics through an HTTP server. The values of the metrics exposed through the HTTP endpoint will be updated as new data is received from Apache Kafka.

  The details of the generated metrics (e.g., name, type, or label names), as well as the URL to the exposed HTTP endpoint, are additional metadata to be captured by the Scorpio broker.

- **Prometheus Pushgateway**

  For those situations in which metrics cannot be scraped, it is also possible pushing metrics to an intermediate component called the Pushgateway. The Prometheus Pushgateway stores those pushed metrics for short period of time and exposes them through an HTTP endpoint that Prometheus can scrape.

  This method is less performant when compared to the exporter, but its implementation is straightforward. Coming back to Apache NiFi, the implemented flow would look similar to the above proposed for the exporter, but instead of exposing an HTTP endpoint, in this case the encoded metrics would be sent straight to the Prometheus Pushgateway.

  The metadata associated to this workflow will also describe details of the Pushgateway such as the URL or the version. This information will be consumed by Weaver in order to specify the destination of the metrics sent from Apache NiFi.

2.2. Available Tools

This section summarizes the tools that have been adopted for the KPI evaluation at the different pilots. Starting from D4.2, this section has been designed to provide a description of the tools, following their incremental evolution during the project.

Considering the tools already described in D4.2 (i.e., the End-to-end Unidirectional Link Latency Evaluator, the Prometheus node exporter, the Prometheus Blackbox exporter, the ELK stack, the 5Probe, the Data Shipper, the Commercial network testing tools and the additional tools integrated

with 5Growth stack), no upgrades/enhancement have been developed. For more details on those tools we refer the reader to the mentioned deliverable.

However, two additional tools have been considered for the KPI evaluation: the log management tool for 5Growth service orchestration and the Passive Network Interface Probe. The details of the tools are reported in the following sections.

2.2.1. Log Management Tool for 5Growth Service Orchestration

Figure 4 shows the general architecture for the integration of log-parser in an end-to-end mobile network management architecture of 5Growth including the demonstration setup and workflow structure. This architecture is a data pipeline that manages the monitoring of instantiation, scaling, and termination time-related metrics from network services initiated by the 5Growth Service Orchestrator (5Gr-SO) in both batch and real-time mode, collecting log changes inside the 5Gr-SO. To build the data engineering pipeline, the whole architecture is composed of five main modules as shown in Figure 5: (a) Data Connection; (b) Data Ingestion; (c) Data Pre-processing; (d) Data Visualization; and (e) Data Analysis.

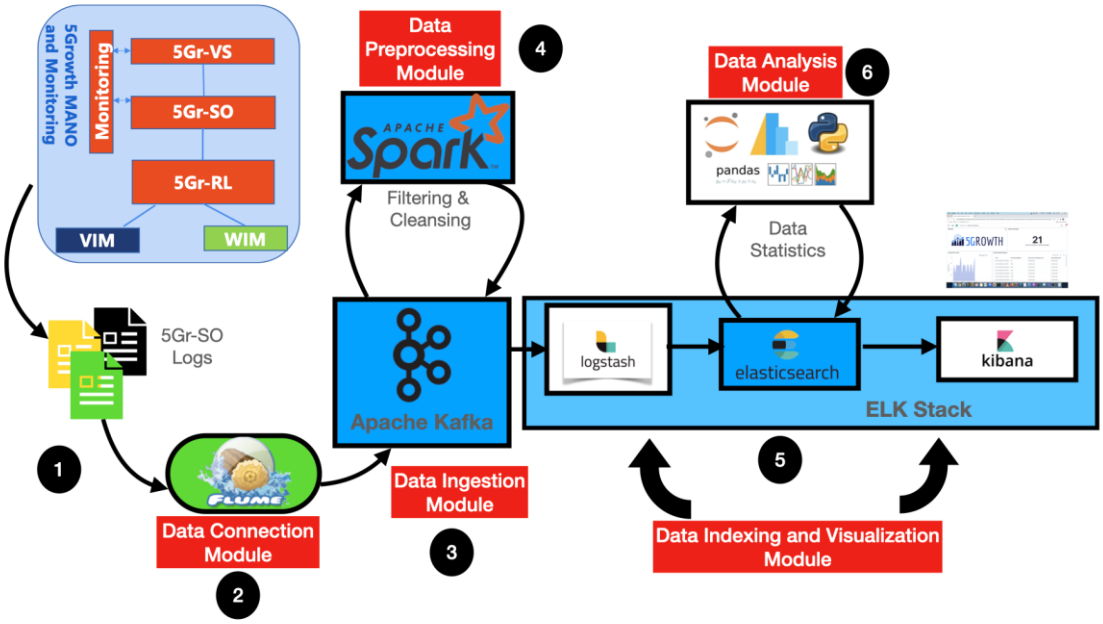

FIGURE 4: GENERAL ARCHITECTURE FOR LOG MANAGAMENT TOOL FOR 5GROWTH SERVICE ORCHESTRATOR

In the Data Connection module, Apache Flume[1] is used for data connection. First, the 5Gr-SO starts logging the whole service instantiation (termination or scaling) process in step (1). Flume application periodically monitors the changes in the logs of the 5Gr-SO as given in step (2) and sends data to

[1] https://flume.apache.org/, Accessed: May-2021

the data ingestion layer. In the Data Ingestion module, the Apache Kafka[2] is used. The Kafka application ingests log messages transmitted from Apache Flume and the log messages fetched by Kafka are temporarily stored in the Kafka broker under a given topic name as given in step (3). In Data Pre-processing module, Apache Spark[3] is used for cleansing real-time data. The main task of the Spark job is to parse and cleanse the incoming log messages from the Kafka topic and to extract the relevant metric values by using Spark libraries in real-time. Every predefined duration of time, e.g., 1 second, a Spark job in the data analysis and processing manager of the data pipeline, collects data from the Kafka broker and cleanses the log data. After obtaining the enriched and cleansed results of the Spark job, they are sent back to the Kafka broker so that the results can be picked up as given in step (4). Finally, Elasticsearch (ELK) stack[4] is used in the Data Visualization module. In step (5), data in Kafka is collected by the Logstash module and is pushed to Elasticsearch so that the output of log time differences of the system can be visualized by Kibana. The values of the calculated metrics are presented in the dashboard screen of Kibana. Elasticsearch is using Logstash subscribed to the Kafka topic to gather the metrics. The dashboard shown in Figure 5 displays an example of monitoring metrics of the network instantiation operation. It can display the number of processed Kafka topics in the data engineering pipeline by Apache Spark and the results of the service instantiation time values of each metric. For Data Analysis, Python libraries are used (e.g., sklearn, pandas, and seaborn) to statistically analyze the data that is indexed by Elasticsearch via a connection to the search engine using the Python Elasticsearch Client[5]. The observed metrics of log data in ELK stack is statistically analyzed in step (6).

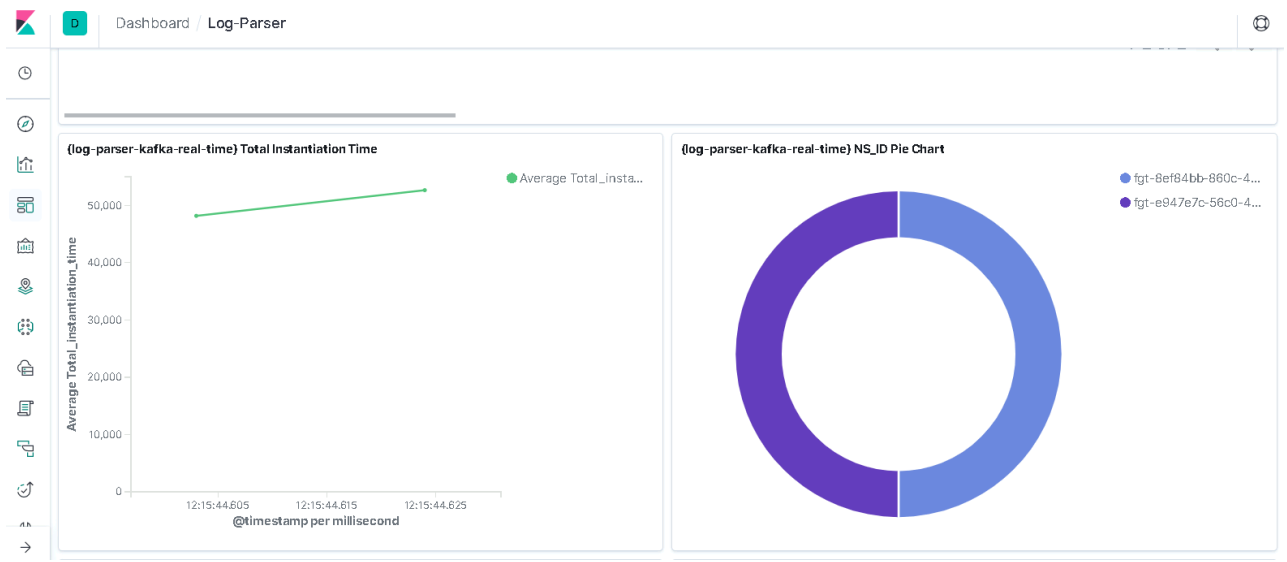

FIGURE 5: KIBANA DASHBOARD FOR 5GR-SO LOG MONITORING

[2] https://kafka.apache.org/, Accessed: May-2021
[3] https://spark.apache.org/ , Accessed: May-2021
[4] https://www.elastic.co/, Accessed: May-2021
[5] https://elasticsearch-py.readthedocs.io/en/v7.12.1/ , Accessed: April-2021

The 5Gr-SO-related log files can be parsed to calculate the various time difference metrics during the network service instantiation operation as given in Table 2 during the scaling operation as given Table 3, and during the termination operation as given in Table 4.

TABLE 2: DATA MODEL FOR 5GR-SO RELATED METRICS FOR INSTANTIATION OPERATION

Metric Name	Data Type	Description
Total_instantiation_time	Long	The time it takes since the 5Gr-SO created the service identifier for a network service until it has been totally instantiated
NS_ID	String	Network Service Identifier (ID)
NSD_ID	String	Network Services Descriptor ID
Operation	String	Identifier of the type of operation: "Instantiation"
Operation_ID_for_Instantiation_Op	Long	The time it takes the Northbound Interface (NBI) to generate an ID to identify the instantiation operation
Hierarchical_SOE_dispatching	Long	The time the hierarchical Service Orchestrator Engine (SOE) uses to select the appropriate instantiation process based on the nature of the service (single NS, composite NS)
Retrieving_descriptor_from_catalogue_DBs	Long	The time to collect the descriptor from the network service descriptor (NSD) catalogue
ROE_parsing_NSDs	Long	The time at Resource Orchestration Engine (ROE) submodule to parse NSD and VNFDs of a network service to get the required information for Placement Algorithm (PA)
ROE_retrieve_RL_resources	Long	The time to recollect the information from the Resource Layer (RL)
PA_calculation	Long	The time to build the request to the PA, send it to the external PA service, and receive its answers
ROE_extract_VLs	Long	The time it takes the ROE to determine the request of the different VLs needing resources in the Logical links (LLs) because connected VNFs have been deployed in multiple Virtualized Infrastructure Managers (VIMs)
ROE_created_VLs	Long	Time in the interaction between ROE and RL to allocate resources in the LLs based on the ROE extract request
ROE_updating_DBs	Long	Time to update DBs to declare the NS as operative and the instantiation operation as successful
Create_monitoring_jobs	Long	Time interaction between SOE and Monitoring Manager modules of 5Gr-SO to determine the monitoring jobs (exporters) and dashboards to be configured at the 5Gr-VoMS plus the interaction to configure them and receive the associated object

		identifiers and update the information in the Network Service Instantiation Resource (NSIR) DB
Create_threshold_based_alerts	Long	Time interaction between SOE-Service Level Agreement (SLA) Manager modules of the 5Gr-SO to determine the threshold-alerts objects (if there is not AI/ML treatment) to be configured at the 5Gr-VoMS plus the interaction to configure them at the 5GR-VoMs and receive the associated object identifiers and update the information in NSIR DB
Create_AIML_alerts	Long	Time interaction between SOE-SLA Manager to configure the AI/ML workflow to drive scaling operations. The creation and configuration of the data engineering pipeline consist of: i) interaction with 5Gr-VoMs to create a Kafka Topic, ii) interaction with the 5Gr-AIML platform to download the required model, iii) creation of inference job at Apache Spark, iv) update of NSIR DB
SOE_time	Long	Time spent in the SOE module (both at SOE parent, SOE child sub-modules) during the instantiation process
ROE_time	Long	Time spent in the ROE module during the instantiation process
CoreMANO_Wrapper_time	Long	Time spent in the Core MANO Wrapper module during the instantiation process to create virtual network supporting the VLs, the VMs supporting the VNFs, and update the NSIR DB
Current_time	Date	Date and time in which the time-related metric values have been sent to Kafka

TABLE 3: DATA MODEL FOR 5GR-SO RELATED METRICS FOR SCALING OPERATION

Metric Name	Data Type	Description
Total_scaling_time	Long	The time it takes to the 5Gr-SO to perform the scaling operation since the request arrives to the NBI
NS_ID	String	Network Service ID
NSD_ID	String	Network Services Descriptor ID
Operation	String	Identifier of the type of operation: "Scaling"
Operation_ID_for_Scaling_Op	Long	The time it takes the NBI to generate an ID to identify the scaling operation
Hierarchical_SOE_dispatching	Long	The time the hierarchical SOE uses to select the appropriate scaling process based on the nature of the service (single NS, composite NS)
SOEc_preparing_info_for_scaling	Long	The time to collect the descriptor from the NSD catalogue

ROE_checking_resource_availability	Long	The time the ROE requires to detect the changes in the NS due to the change in instantiation level and determining if there are enough resources in VIMs to satisfy the scaling request. During this process, it also interacts with the RL to get the resources
ROE_extracted_scaled_VLs_at_RL	Long	Time at the ROE to determine the request to establish the new created/deleted networking resources at the LLs based on the new created/deleted VNFs and the rest of the service
ROE_scaled_LLs_at_RL	Long	Interaction with the RL to create/remove the resources determined in previous row
ROE_updated_dBs	Long	Time to update DBs to declare the NS as operative and the scaling operation as successful
Scale_monitoring_jobs	Long	Time to determine the monitoring jobs (exporters) and dashboards to be configured at the 5Gr-VoMS plus the interaction to configure them and receive the associated object identifiers and update the information in NSIR DB
Scale_AIML_jobs	Long	Time it takes the SOE to interact with the SLA manager to create a new Inference Job (Spark streaming) after scaling to process information
SOE_time	Long	Time spent in the SOE module during the scaling process
ROE_time	Long	Time spent in the ROE module during the scaling process
CoreMANO_Wrapper_time	Long	Time spent in the Core MANO Wrapper module during the scaling process to determine the operations needed to pass from current to target Instantiation Level (IL), create/delete the VMs associated to the VNF instances, and update the NSIR DB
Current_time	Date	Date and time in which the time-related metric values have been sent to Kafka

TABLE 4: DATA MODEL FOR 5GR-SO RELATED METRICS FOR TERMINATION OPERATION

Metric Name	Data Type	Description
Total_termination_time	Long	The time it takes to the 5Gr-SO to perform the termination operation since the request arrives to the NBI
NS_ID	String	Network Service ID
NSD_ID	String	Network Services Descriptor ID
Operation	String	Identifier of the type of operation: "Termination"
Operation_ID_for_Termination_Op	Long	The time it takes the NBI to generate an ID to identify the termination operation

Hierarchical_SOE_dispatching	Long	The time the hierarchical Service Orchestration Engine (SOE) uses to select the appropriate terminating process based on the nature of the service (single NS, composite NS)
Terminating_AIML_alert_jobs	Long	The time required in the interaction between SOE and SLA Manager to remove data engineering pipeline elements (Kafka topic, Spark job) in case they have been configured in the 5Gr-VoMs and the Inference Platform (Apache Spark)
Terminating_Threshold-based_alerts	Long	The time required in the interaction between SOE and SLA Manager to remove alert-based objects in case they have been configured in the 5Gr-VoMS
Terminating_Monitoring_jobs	Long	The time required in the interaction between SOE and Monitoring Manager to remove alert-based objects in case they have been configured in the 5Gr-VoMS
ROE_deleting_LLs	Long	Time in the ROE to interact with the RL to deallocate resources in the LLs serving the VL connections between VNFs deployed in multiple VIMs
ROE_updating_DBs	Long	Time to update DBs to declare the NS as operative and the termination operation as successful
SOE_time	Long	Time spent in the SOE module during the termination process
ROE_time	Long	Time spent in the ROE module during the termination process
CoreMANO_Wrapper_time	Long	Time spent in the Core MANO Wrapper module during the termination process to delete virtual network supporting the VLs and the VMs supporting the VNFs
Current_time	Date	Date and time in which the time-related metric values have been sent to Kafka

2.2.2. Passive Network Interface Probe

This tool is a straightforward passive network interface probe. It has been designed to report the average throughput (considering both packet per second and bits per second) in each direction of a Virtual Machine (VM) hosted within a hypervisor or a CPE, using a Prometheus Pushgateway for the data exporting (compliant with the 5Gr-VoMS).

The probe uses the network interface statistics gathered by the Operating System's (OS) kernel via the *psutil*[6] Python module, allowing for more extensive compatibility with the different OS types available within the CPEs.

[6] https://github.com/giampaolo/psutil

When running within a *qemu+KVM* hypervisor environment, such as an OpenStack deployment, the probe gathers the statistics straight from the TAP interfaces exposed in the compute node that hosts that VM. Because of this, the upload of the hypervisor is the download of the VM (and vice-versa). Therefore, the probe has different code paths for hypervisors and CPEs that automatically inverts the send and receive direction, when needed. Furthermore, the metrics are always reported to evaluate a vertical service. Thus, when monitoring VMs from the hypervisor, the reported metrics will be consistent with those measured within a CPE.

2.3. Experiment Catalogue

The 5Growth experiment catalogue consist of the result different experiments conducted by in the 5Growth pilot sites, as well as associated metadata. More precisely, the primary goal of the experiment catalogue is to enable the reproducibility of experimental results. To that end, the experimental results are paired with additional information – the experiment descriptor. This section first outlines the requirements defining their content (i.e., what needs to be registered alongside experimental results to fully capture its context and operations), and then outlines the high-level steps followed by experiments from which the content of experiment descriptors can be assembled.

2.3.1. Required Experimental Context

To capture the whole context of a given experiment, an experiment descriptor should answer three high-level questions:

1. **What does the experiment measure?** The experiment descriptor must precisely capture the final SKPI to be measured by the experiment, as well as its eventual decomposition into CKPIs, and the corresponding aggregation functions. These CKPIs must be then paired with specific measurement tools, known to be usable on the target testbed.
2. **How are tools used?** The experiment descriptor must extensively describe each tool in use. Starting from a description of the tool itself (e.g., version information if it matters, provenance – compiled from source with specific flags, part of a given version of an Operating System (OS) stack, from a package manager), then covering their use either manually or scripted (e.g., command-line arguments, configuration files, start/stop commands), and finally all necessary metadata about the results the tool produce (how to interpret it, corresponding data consumer).
3. **Under what experimental conditions?** The experiment took place on a given network infrastructure (e.g., topology, overlays), at specific moments in time. Furthermore, the experiment may be conducted in isolation (either through a slice or on an idle network), as opposed to on a "live" network, with its usual/expected production loads, and/or the experiment required to generate specific background traffic patterns.

An entry in the experiment catalogue, specific to a given testbed, is then the tuple formed by a given experiment descriptor and its set of results.

2.3.2. Deriving Experiment Descriptor Content

Results from a single experiment can come from a variety of data sources (e.g., tools in use, network nodes on which they are deployed, measured metrics), with specific temporalities (e.g., sampling frequencies and/or conditions), and be subject to externals factors (e.g., concurrent production traffic, and/or synthetic background load with a well-defined pattern). Finally, they must be post-processed, possibly several times, to produce the measurement value for the target (S)KPI. All these factors can influence the result of an experiment and thus, need to be recorded inside the experiment catalogue entries, i.e., the experiment descriptors. Besides enabling the results reproducibility, this also ensures that the results can be properly compared. This allows comparing the influence of some factors such as the impact of background traffic, or diagnosing why results differ. While the actual format of the experiment descriptors will vary depending on the underlying platform (e.g., 5G EVE testbeds use JSON schemas, other might use a textual description and/or scripts), their content should be equivalent.

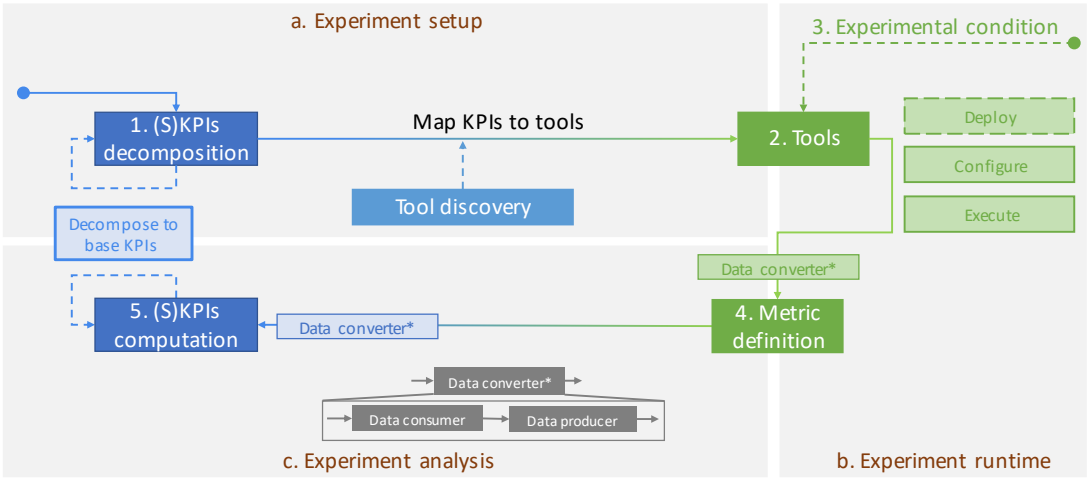

FIGURE 6: EXPERIMENTS TRANSITIONS THROUGH 3 MAIN PHASES, EACH INTRODUCING ADDITIONAL SOURCES OF CONTEXT ABLE TO INFLUENCE THE RESULTS OF AN EXPERIMENT

Figure 6 illustrates the three main phases through which an experiment transitions, before eventually outputting its result. Experiment descriptors must provide the required information to perform these different phases, thus must capture the corresponding context sources that are introduced by each phase and will influence the value of the result. We now detail the main five context sources:

1. **KPI decomposition:** 5Growth experiments aim to measure and validate SKPIs by measuring and compounding CKPIs as described in the deliverable D4.2. This initial KPI decomposition step is thus the first possible source of variance across multiple executions of an experiment. For example, depending on the underlying service architecture (e.g., video codec, transport protocol), the P1UC1-SKPI-2: High-resolution real-time video quality, as measured in Section 4.1.1 may or may not need to consider the end-to-end latency as CKPI if the service is operating over a lossy network.
2. **Measurement tools:** A given CKPI can be measured in many different ways, through the use of different tools and/or post-processing steps. As such, it is crucial to accurately record which tools were used to measure which KPI, but also more importantly how those tools were used

(e.g., where in the network, how they were built/deployed/configured/executed, what was the host system configuration). More importantly, the availability of some tools will vary depending on the experiment site/configuration, impacting both the ability to measure a given KPI, and the reported results.

For example, traffic generators can be used to inject artificial traffic in an experiment. Their generated traffic patterns can however vary largely depending on multiple factors. E.g., where were the client/server deployed, what was the underlying transport protocol (e.g., congestion control algorithm, use of pacing), or how many parallel flows were active at the same time. Finally, different traffic generators will support different ways to measure the same given metric, which may lead to collect different results (e.g., estimating the throughput using the TCP congestion window/smoothed RTT or by averaging the amount of delivered data over the run can give different throughput estimation over bursty access technologies).

3. **Experimental conditions:** External factors can play a major role in experiment results. More precisely, we distinguish between two sets of external factors, making up the experimental conditions during which an experiment was carried out: (i) the configuration of the infrastructure (both its network properties, as well as the individual nodes properties); and (ii) the status of the network during the experiment runtime. For example, the specific slice (and the associated SLAs) in which an experiment ran, as well as the OS version and configuration of the nodes used to run the testing tools (e.g., socket memory limits) capture infrastructure properties relevant for an experiment using a traffic generator. Recording whether the experiment ran alongside production traffic, and when/how often, alongside the topologies of the network/slice/service capture its runtime properties.
4. **Metric definition:** Properly interpreting measurement results requires some context on how these were computed in the first place, and what they mean. This is relevant to enable the ingestion of the results into the 5Growth data infrastructure platform through data producers, as these need to advertise the data they generate using a common representation (see Section 3.1 of the deliverable D4.2 .

   For example, key properties to track are the measurement units, their periodicity if applicable, whether these represent aggregates/samples or are "raw", and the eventual transformation/post-processing that might be needed to apply on the tool output to make it processable by the data infrastructure platform.
5. **KPI computation** Similarly to the result collection, the computations performed to transforms raw metrics results into CKPIs, and the subsequent recombination into SKPIs are key to understand the results of an experiment, compare them across experiments, and/or reproduce them.

   For example, averaging jitter measurements as opposed to quantizing them and looking at the resulting P99 values is likely to lead to different 5GR-SKPI-4-High-resolution-Real-time-Video-Quality results if the network is fully utilized.

Analysing all these experiment phases and context sources enable to assemble an **experiment descriptor**, which captures all factors influencing the experiment and defining its runtime

environment. A formal description of these descriptors, leveraging the ones originally derived for the 5G EVE testbed, is presented in the next section.

2.4. Experiment Descriptors

The experiment descriptor defines all the information required to deploy, test, and validate a vertical service over a 5G infrastructure. This descriptor provides a common template, which includes elements to describe the service deployment in the target testing environment, the metrics to be collected and the mechanisms and criteria to compute, elaborate, and evaluate the KPIs during the Pilots validation campaigns. In this sense, the 5Growth Experiment Catalogue described in Section 2.3 contains the motivation and rationale to have a 5Growth experiment catalogue, and the information required to describe a 5Growth experiment. This section describes the Information Models (IMs) of the 5Growth Experiment Descriptors, which are based on an evolution of the Experiment Blueprints defined in the 5G EVE project, with minor adaptations introduced to cover 5Growth-specific requirements.

The selection of the 5G EVE Experiment Blueprint information model as baseline for 5Growth Experiment Descriptors is motivated by the following considerations: (i) the approach and the information models developed in 5G EVE fulfil the objectives established in Section 2.3; (ii) two 5Growth Pilots, namely INNOVALIA and COMAU, leverage the 5G EVE testing facilities for the pilot campaign and therefore 5G EVE blueprints are implicitly used to model (at least part of) the services and the experiments for these two pilots; (iii) both 5G EVE and 5Growth projects use blueprints and descriptors evolved from the Vertical Service Blueprints originally defined in the 5G-TRANSFORMER project, and therefore share the same conceptual approach and part of the information models, which will ease the development of the experiment descriptors.

As described in [17], 5G EVE uses four main information elements to enable a flexible definition and customization of an experiment using high-level information:

- **Vertical Service Blueprint (VSB):** 5G EVE adopts the same VSB concept used in 5Growth to model the vertical service to be tested during the experiment execution. This is a high-level template describing the components of a vertical application and their interconnection.
- **Context Blueprint (CB)**: a high-level template describing the elements injected in the experiment deployment to emulate certain operational condition. This operational condition aims to model the environment where a vertical service instance may be deployed in a realistic scenario. For example, a CB may describe elements introduced to generate background traffic or delays in certain links of the vertical application.
- **Test Case Blueprint (TCB**): a template defining the scripts and the variables for the automatic configuration and execution of the test cases, which are automatically managed through the 5G EVE platform.
- **Experiment Blueprint (ExpB)**: a high-level template that composes the vertical service blueprints, context blueprints and test case blueprints, in order to define the complete experiment deployment and the steps for its automated execution and validation. This

includes the definition of the KPIs in terms of the metrics provided by the different components.

The 5G EVE IMs were designed to offer a flexible composition of the different elements, to allow experimenters dynamic mechanisms to test automatically the vertical service applications under different operational conditions. The Experiment Descriptors in 5Growth, however, aim to model the specific deployment and operational context used during the validation, but they do not have the objective of automating the test execution and validation, which is supposed to be performed manually. Therefore, the modifications were made to adapt the 5G EVE descriptors to allow to describe a concrete experiment scenario in a user-friendly manner. In particular, while the 5G EVE platform manages the automated experiment execution through the Ansible scripts defined in the TCBs, in 5Growth the testing procedures are modelled through a human-readable description of the steps to be performed during the validation campaign for the experiment configuration and execution. In other terms, instead of using the concept of Test Case Blueprint, the 5Growth Experiment Descriptor includes a textual test execution description. The resulting information model is shown in Figure 7.

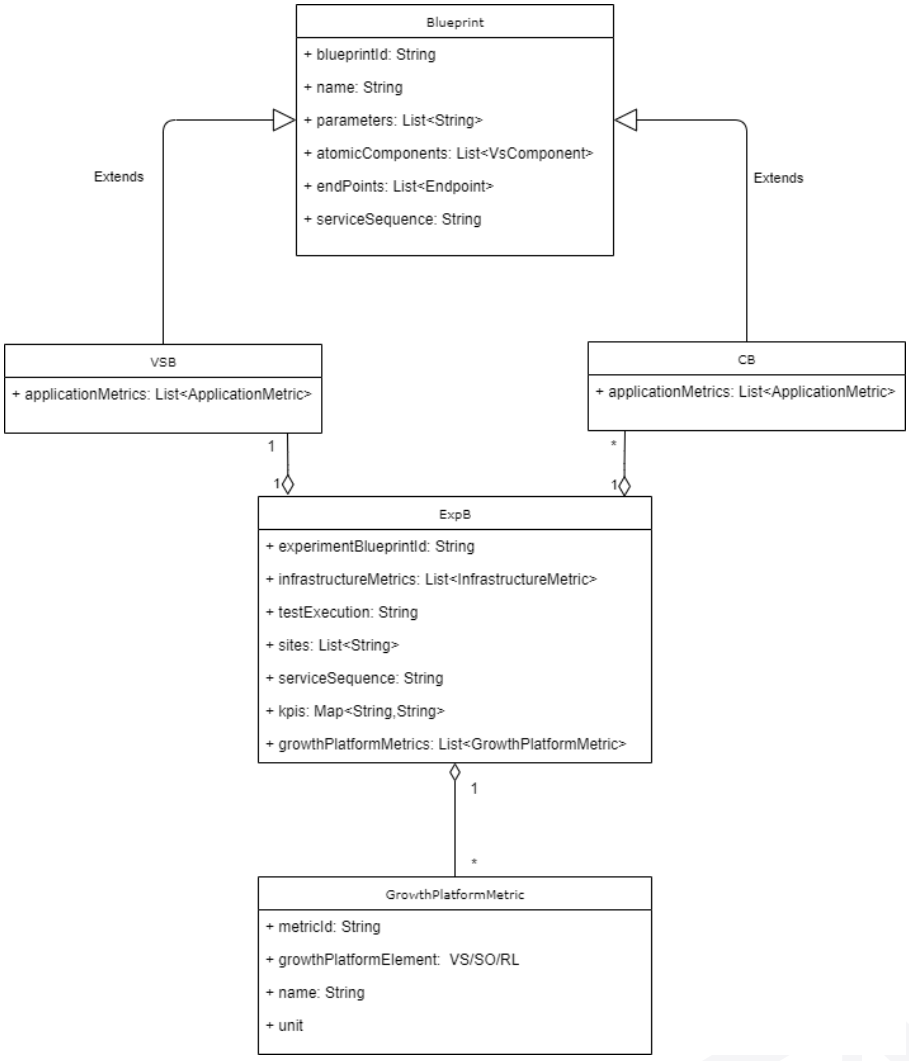

FIGURE 7: 5GROWTH EXPERIMENT DESCRIPTOR INFORMATION ELEMENTS AND MODELS

Vertical Service Blueprints (VSB) and Context Blueprint (CB) extend the base *Blueprint* class, which corresponds to a 5Growth VSB. To define experiments, the *VSB* and *CB* extend the IM of the base *Blueprint*, adding the list detailing the metrics the vertical service or the experiment context elements generate for the KPI calculation (i.e., the *applicationMetrics* field). We refer to [17] for further details about the *ApplicationMetric* and *InfrastructureMetric* definition, and to [16] for further details regarding the Blueprint parameters.

The *ExpB* is composed of an instance of VSB and multiple instances of CB and includes further parameters to identify the different types of metrics to be collected, the criteria to compute and evaluate the KPIs and the steps to execute the test. In details, the metrics may include metrics coming from the infrastructure (e.g., Radio uplink data rates, packet losses, etc.), or metrics that are produced by the 5Growth Platform itself (e.g., Service instantiation time) i.e., the *infrastructureMetrics* and the *growthPlatformMetrics* respectively. We refer to [17] for further details regarding the *infrastructureMetrics*. The *kpis* field allows to determine how each of the Pilot KPIs can be extracted from the metrics, using a human-friendly description referencing the different metrics described by the blueprints. The *testExecution* field defines, in a human-friendly manner, the steps to be performed for configuring and executing the scenario for the Pilot validation. Finally, the *sites* field allows to specify the target test infrastructure where the experiment must be instantiated during the validation campaign.

3. Pilot Integration

This section presents considerations on the metrics and technical requirements relevant for each of the four pilots, along with the measurement methodology therein. This is followed by a detailed description on the integration of monitoring and other tools from ICT-17 platforms and 5Growth. Finally, the document ends with the presentation on how each pilot's selected WP2 innovations are integrated into its operation.

3.1. INNOVALIA Pilot

3.1.1. Technical Requirements and Related KPIs

There are some considerations to be considered concerning the metrics that are relevant for this pilot, in terms of technical constraints and requirements from the vertical.

In terms of user-plane latency (CKPI-1 End-to-end Latency), there are several segments in the path of the traffic that contribute for the overall latency. There are network related latencies that will impact all kind of traffic. The one-way delay values introduced would be: up to 10 ms on a first RAN segment; half a ms per 100 km on the transport segment; close to 0 between the edge and the RAN; and up to 10 ms on a second RAN segment. Also, there are application-level latencies that will have a different impact on the overall latency depending on the application. When experimenting the INNOVALIA application services with no distance between sites, the minimum latency value for the service is obtained.

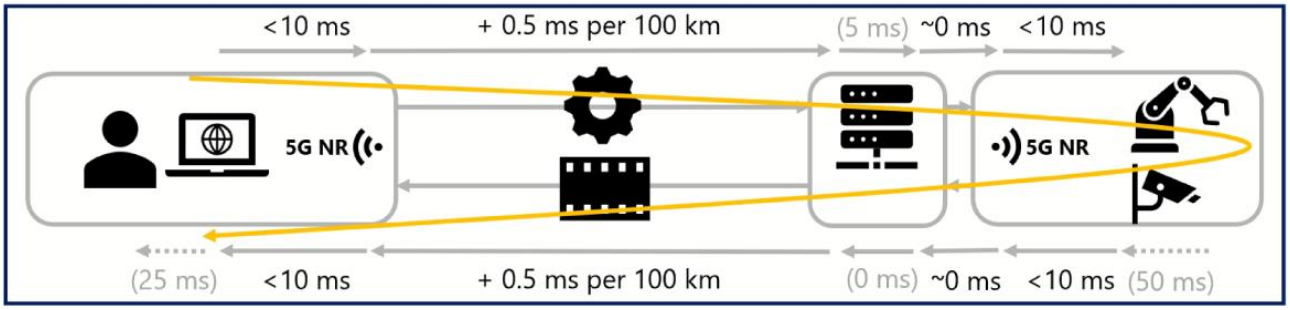

FIGURE 8: RTT LATENCY

According to the ITU-R M.2410-0 "Minimum requirements related to technical performance for IMT-2020 radio interfaces" [18], the one-way delay on the RAN segment for an enhanced Mobile Broadband (eMBB) type communication should be 4 ms with small IP packets. Therefore, the one-way delay on the RAN with real TCP traffic is in the range of 5-10 ms.

In this pilot, it is being verified which would be the maximum distance between sites that would still allow the operation of the service. The distance between sites is being emulated by adding artificial latency in steps of 5 ms on each way.

Regarding packet loss rate (CKPI-2), it is assumed in this pilot that the value should be below 2%, which is a value that would not have a high impact on the quality of the service and a value that a commercial 5G network offered by a Customer Service Provider should be able to guarantee.

As for traffic data rate (CKPI-3 Guaranteed Data Rate), it has been evidenced that the demand from the streaming video reaches around 22-24 Mbps, while the demand from the CMM traffic peaks to 13-15 Mbps, but the latter is not constant traffic, it has an irregular pattern. So, the total peak demand would be in any case under 40 Mbps, which is well met by the 5G radio resources available and configuration implemented at 5TONIC. At this lab, the implemented conditions are: 50 MHz bandwidth within the 3.5 GHz band, a TDD pattern of 7:3, 256 QAM modulation and 4x4 MIMO. With such radio conditions, the maximum achieved data rate is 544 Mbps in the downlink and 54 Mbps in the uplink.

With regard to availability (CKPI-5), it is being assumed a value over 98% in this lab environment, which provides more than enough availability of the components for the vertical services to be tested. In a production commercial environment, the availability would be even higher as those deployments are implemented with higher redundancy and Customer Service Providers aim to guarantee the "five nines" (99.999%) of availability.

About the slice creation time (CKPI-6), the requirement from the vertical is to be in the order of a few minutes for the instantiation of the service (a value of 5 minutes is set as a more specific target, which would be good enough for the vertical and technically feasible). This time refers to the time since the service deployment is requested until the service is fully available. It includes the time of instantiation of the VMs, which is the task that takes longer in the process. There are no more than 2 VMs in the Network Services used in this pilot, so it does not take too long to launch, and the requirement is met.

All this being said, the requirements established for INNOVALIA use cases in the project proposal are updated, in accordance with the considerations and evidence described above, and shown in Table 5.

TABLE 5: 5G REQUIREMENTS FOR INNOVALIA USE CASES

	INNOVALIA UC1	INNOVALIA UC2
Network Latency	<20 ms + 0.5 ms/100 km	<20 ms
Data Rate	40 Mbps	40 Mbps
Mobility	static	3-50 km/h

The other KPIs (i.e., Availability, Connection Density and Wide-Area Coverage) included in the original table cannot be measured within this pilot. A very high Availability is something that will be provided in a highly redundant commercial network deployed by a Customer Service Provider. The use case setups evaluated in this pilot do not involve a big number of devices, so the Connection Density is not a concern in this pilot. The equipment and setup needed for the use cases just take a small area, thus the Wide Area Coverage is not actually a requirement for this pilot.

There was also a mention to the 5G-PPP KPI objectives in the project proposal. Those are shown in Table 6 below:

TABLE 6: 5G-PPP OBJETIVES APPLIED TO INNOVALIA PILOT

Vertical Pilot 1: Industry 4.0	• Provide 1000 times higher Wireless area capacity and more diverse service capabilities compared to 2010. • Create a secure, reliable and dependable Internet with a "zero perceived" downtime for services provision.
Vertical Pilot 2: Industry 4.0	• Provide 1000 times higher Wireless area capacity and more diverse service capabilities compared to 2010. • Create a secure, reliable and dependable Internet with a "zero perceived" downtime for services provision.

The 1000 times higher wireless area capacity could only be achieved by using all the bandwidth from all the frequency bands that will be assigned for 5G, which cannot be tested in the scope of this project. There will be spectrum assigned for 5G in low-bands (700 MHz band), in mid-bands (3.5 GHz band) and in high-bands (28 GHz band, and also called mmWave). The low-band has not been yet released and authorized for 5G use yet, there are other technologies that are using this band currently. So far, in this pilot it has only been tested the mid-band, using the 50 MHz bandwidth authorized for Telefonica in Spain. The high-band will be tested in the last experimentation phase at Bilbao. In the high-band, there is more bandwidth available than in the other bands, so higher data rates could be reached if demanded by the vertical service. Anyway, as seen during the experimentation performed in this pilot to this day, higher data rates have not been required.

The zero perceived downtime can only be tested on a stable highly redundant network and keeping it monitored for months. In a development environment, where new implementations are being done constantly, new features are being developed and upgrades are being performed often, this cannot be measured.

3.1.2. Measurement Procedures

This information is already detailed in D4.2 section 2.2.1. No modifications have been done in this regard since the delivery of D4.2.

3.1.3. ICT-17 and 5Growth Platform Integration

This section is only including the integration of the monitoring part, as other features have already been described in previous deliverables D3.2, D3.3 and D3.4. In the case of the INNOVALIA pilot, this integration process is oriented to collect the monitoring data generated in the 5G EVE experiments, in order to make it available for the 5Growth Monitoring platform. This monitoring data is generated by the 5Probe (see description in D4.2 Section 2.3.5) deployed in 5TONIC for collecting metrics related to the network infrastructure.

This requires the adaptation of some components in both 5G EVE and 5Growth stacks in order to enable this interaction, because the information model followed by the monitoring data in 5G EVE [17] is not compatible with the information model managed by the 5Growth Monitoring platform,

so that the aggregation of the monitoring data from the 5G EVE Monitoring platform and its delivery to the 5Growth Monitoring platform is needed. For this purpose, the Semantic Data Aggregator already presented in Section 2.1.1 is used for managing this process, following the 5G-EVE Kafka broker data source integration already detailed in 2.1.2.

This integration has required the following changes and updates:

- First of all, it is needed to provide a specific use case name in the 5G EVE Portal when declaring the experiment, different than others already used. In this case, the use case will be identified by the keyword "innovalia_5growth". This keyword will be eventually used to retrieve the corresponding Kafka [19] topics by the components in charge of the integration.
- The DCM Site Plugin, a Python script running in the 5G EVE site facilities to manage the lifecycle of the topics installed on each site, has been updated to allow the obtention of the Kafka topics which contain the "innovalia_5growth" keyword. This modification will be only included in the 5TONIC Kafka broker, as 5TONIC is the site facility in which the experiments will be executed and in which the 5Growth platform is deployed.
- The Semantic Data Aggregator, component in charge of translating the 5G EVE data model into the 5Growth data model, needs to retrieve the Kafka topics to which it has to subscribe to by triggering the new endpoint configured in the 5TONIC's DCM Site Plugin. This process is being done manually (i.e., without using automated scripts) for the time being.

As a result, the following high-level workflow is followed to enable this integration:

- First of all, the experiment needs to be deployed in the 5G EVE platform.
- Before starting with the execution of the experiment, moment in which the Kafka topics will be already available, the Semantic Data Aggregator retrieves the topic names related to the "innovalia_5growth" by sending the proper request to the 5TONIC's DCM Site Plugin.
- After obtaining the topics, the Semantic Data Aggregator subscribes to these topics and start consuming data from them.
- Then, it is time to start the execution of the experiment.
- When the experiment starts and the 5Probe starts to publish data to the 5TONIC Kafka broker, the Semantic Data Aggregator will receive the monitoring data, then doing the translation to the 5Growth data model and publishing the data to any entity from the 5Growth platform interested in receiving it.

3.1.4. Innovation Integration

It has been analyzed which of the innovations developed within WP2 could be integrated into the INNOVALIA use cases. Several innovations are not suitable for being integrated in this pilot since some of them do not fit a business motivation or they are not applicable in the way this pilot is implementing the integration with the 5G EVE ICT-17 platform. That said, there are two innovations that are relevant for this pilot: (i) the innovation I6 Federation and Inter-domain, and (ii) the innovation I8 Performance isolation.

The integration of the 5Growth platform with the 5G EVE platform relies on the work done around the innovation I6 Federation and Inter-domain. Features such as the VS level split to allow the definition of a vertical sub-service that is onboarded on 5G EVE side or the multi-domain to support requests to different domains, leveraging on specific drivers are used. These features are more broadly explained in D2.3 Section 3.1.6. Also, the specific details of the integration of 5Growth with 5G EVE platform are thoroughly detailed in D3.2 Section 3.3. From the two alternatives considered within that document, the option 2 was the final choice. In the selected alternative, represented in Figure 33 , the Communication Service Management Function (CSMF) of the 5Growth Vertical Slicer (5Gr-VS) sends a request to the 5G EVE portal to trigger the instantiation of the vertical service. Then, the 5G EVE framework continues the workflow and selects the 5Growth site as the place for instantiating the resources. There is a driver at the 5G EVE Interworking Layer (IWL) to build the proper command towards the 5Growth Service Orchestrator (5Gr-SO).

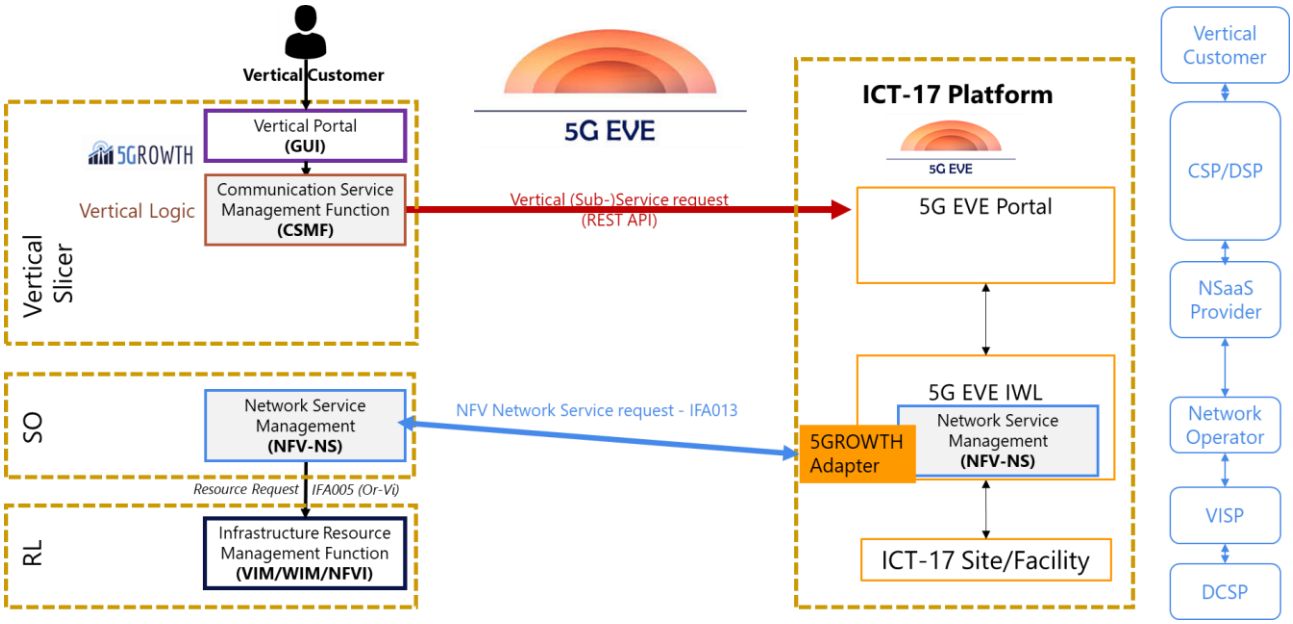

FIGURE 9: 5GROWTH INTEGRATION WITH 5G EVE

The innovation I8 Performance isolation will be further developed as a PoC. The performance isolation innovation allows, by means of virtual queues and policing, prioritizing a critical slice over a non-critical one (in relative terms) deployed over the same network infrastructure. This enables the system operating acceptably even under high-load situations, which increases security, performance, resilience while meeting the expected QoS. This adds an additional level of resilience and reliability for critical UC communications in the Public Network (PN) beyond the capabilities leveraged by the Non-Public Network (NPN) segments of the network in the use case. Further details of this innovation are available in WP2 deliverables.

Within 5Growth, this innovation can be implemented by using ONOS-P4 Slicing App, which allows classifying the traffic in different network slices with different QoS parameters. This includes the application of P4 meters, performing rate-limiting traffic policing, monitoring network traffic for compliance with the maximum rate of traffic received (number of packets per second and maximum burst size) on an interface for a particular network slice and taking steps to enforce that rate. The

performance requirements of a network slice instance can, therefore, be met by making the correct local decision. To do so, new parameters characterizing the slices could be considered for the allocation operation, such as the maximum and minimum bandwidth, the maximum burst size and the Active Queue Management (AQM) parameters for delay guarantees. This operation allows dynamically enforcing the forwarding rules on the data plane devices (i.e., P4 switches) according to the required QoS parameters. *QoSlicing* manages both meters and virtual queues across slices. The performance-isolated network slices can be exposed to verticals across different Points of Presence in the transport network.

The implementation of this innovation implies the deployment of an ONOS-P4 switch between the 5G CPE 2 and the user end devices as shown in Figure 10 below:

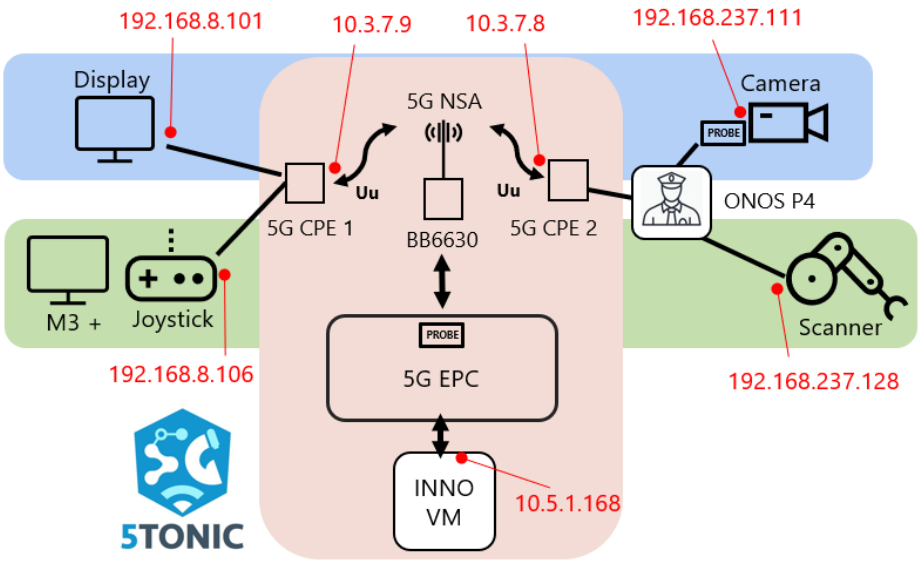

FIGURE 10: SETUP INCLUDING THE ONOS P4 SWITCH

3.2. COMAU Pilot

3.2.1. Technical Requirements and Related KPIs

D1.1 detailed the requirements of the COMAU pilots and of the relevant use cases. Service KPIs, associated with the three use cases of the COMAU pilot, have been finally mapped on Core KPIs which are the entities verified by measurements or qualitative considerations. The association between Service KPIs and Core KPIs are reported in Tables 17/18/19 of D4.2.

As specified in the following, Core-KPIs 1/2/3/9 are measured on the vertical premises. The measurements are done on the radio network combined with the transport network based on optical technologies (a ring of optical fiber). Section 2.2.2. of D4.2 reports the setup used for performing the measurements on both radio and transport segments.

The tests are organized in "sessions" which last at least four days, with the APU measurement boards continuously alternating between iPerf and LaTe measurement sessions. Late (latency tester) is a tool

deployed by POLITO described in Section 2.2.2.1 of D4.2. With the scope to orchestrate the tests, a robust system has been developed. It is also able to manage network and/or power outages (program forking the LaTe and iPerf processes and synchronizing them). More specifically, iPerf is used to measure the throughput in UL and DL, both using UDP and TCP while LaTe is used to assess the RTT latency (and its jitter), DL latency, UL latency, and packet loss.

The Core-KPIs assessed in the tests are the following:

- CKPI-1 End-to-end Latency - This KPI is computed using the LaTe tool, by implementing a client-server loop. The latency in RTT is computed as the time difference between "send timestamps" and "receive timestamps", which are managed by the client and server applications participating in the measurement session. Typically, the "send timestamp" is inserted inside the packet by the client which issues a request and the "receive timestamp" is collected by the client when the reply from the server, containing a copy of the "send timestamp" from the corresponding request, is correctly parsed. Before any session is started, a connection initialization packet is sent by the client and properly acknowledged by the server, to ensure that the measurements can start, and the connection is stable enough. To avoid synchronization issues, packets are sent with a random periodicity, exponentially distributed with minimum value of 40 ms and mean 50 ms. A new value of periodicity is extracted every 10 packets.
- CKPI-2 Packet Loss - the packet loss is measured as a by-product of the latency tests.
- CKPI-3 Guaranteed Data Rate - the data rate is measured by iPerf 2.0.13 in UL and DL, for both TCP and UDP traffic, with a socket application buffer size of 1000 B, to avoid data underflow. The "socket application buffer size" is the size of the application-layer buffer that iPerf sets when using the write() and recv() system calls. The TCP ramp-up transient was removed from the measurements (first two seconds of each test) to focus on the steady state network behavior.
- CKPI-9 Jitter - defined as the variation of latency between packets, the jitter was assessed through several 18-second tests were run and packet latencies were collected using LaTe. For each test interval, the following quantities are computed: (i) the standard deviation of the latency over all the packets in each test, with its 95% confidence intervals; (ii) the difference between the maximum and the minimum latency observed in each test interval.

The remaining CKPIs are verified with alternative methodologies for the reasons stated in the following:

- CKPI-5 Availability – The core KPI related to availability, in relation to the specific service KPIs reported in Tables 17/18/19 of D4.2, refers to the percentage of time during which the radio network ensures the support of the services with the expected QoS. It is the ratio between the up time of the radio link over the total time of the radio link is planned to operate. As the expected availability is in the order of "five-nines", an in-field measurement would require at least one year of continuous test to intercept an "out of order" time with a statistical meaning. In addition, the radio system in the pilot is not a 1+1 (full redundancy) for cost reasons while,

typically, a real deployment would have a full 1+1 redundancy: measurements in the pilot area would bring to availability values poorer than more realistic commercial deployments.

- CKPI-7 Connection Density - The number of devices that can be connected simultaneously to the network infrastructure without degrading the performance of the devices that are already connected can be estimated in 5 devices for each square meter. This is the value specified for the Ericsson commercial radio products used in the area when the mMTC profile is used. The underlying transport network is dimensioned not to limit the radio performances. An in-field verification is not possible because such a density of devices is not present in the pilot.
- CKPI-10 Received Radio Signal Quality – This core KPI is verified at the 5G CPEs level by monitoring the indication of received signal. As the radio antenna is located indoor, in a vicinity of the terminals and without obstacles, it can be argued that the quality is excellent as it is achieved in the best possible conditions. In a realistic deployment, the radio signal quality is achieved by an accurate planning of the antenna footprint in relation to the characteristics of the area.
- CKPI-11 Buffer Occupancy – This core KPIs amount of user-data pending transmission (i.e., submitted by applications and waiting in transmission buffers). This KPI is associated to the UC in relation to the network support for mobility in the factory area. The same consideration of CKPI-10 appliers here: in a realistic deployment, the radio signal quality is achieved by an accurate planning of the antenna footprint in relation to the characteristics of the area and the network is well-dimensioned to avoid bottlenecks. However, this KPI will be validated at the end of the project, when the UC3 will be completely deployed by verifying that the video streaming does not experience lags or delays.

The project proposal (Table 1-1 where this pilot is indicated as "Vertical pilot 2") also reported a mention to the contribution of this pilot to the 5G-PPP KPI objectives. Specifically, this pilot was expected to contribute to:

- Provide 1000 times higher wireless area capacity and more diverse service capabilities compared to 2010.
- Creating a secure, reliable, and dependable Internet with a "zero perceived" downtime for services provision.

The INNOVALIA pilot, reported in Section 3.1 of this document, and the COMAU pilot contribute to these goals in the same way as both target the evolution of smart manufacturing (i.e., Industry 4.0) so the considerations reported in Section 3.1 for INNOVALIA also applies to the COMAU pilot.

3.2.2. Measurement Procedures

This information is already detailed in D4.2 section 2.2.2.

3.2.3. ICT-17 and 5Growth Platform Integration

To perform measurements without physically entering the vertical premises, the exploitation of the monitoring platform of the 5Growth platform is planned. A sample deployment is the one depicted in Figure 11. As shown in figure, one monitoring probe is deployed in a VM/container running in a PC connected to the Customer Premises Equipment (CPE) where user's terminals (e.g. robots, sensors, AR glasses…) are connected. The other probe is deployed in a VM/container in the APP SERVER. The probes are the ones described in D4.2 and deployable through the 5Growth monitoring platform. They can collect latency, jitter, packet loss, the percentage of packet loss and the total transmitted packets. A common or two separate Prometheus exporters then export the data to the 5Growth platform where the monitoring platform is running.

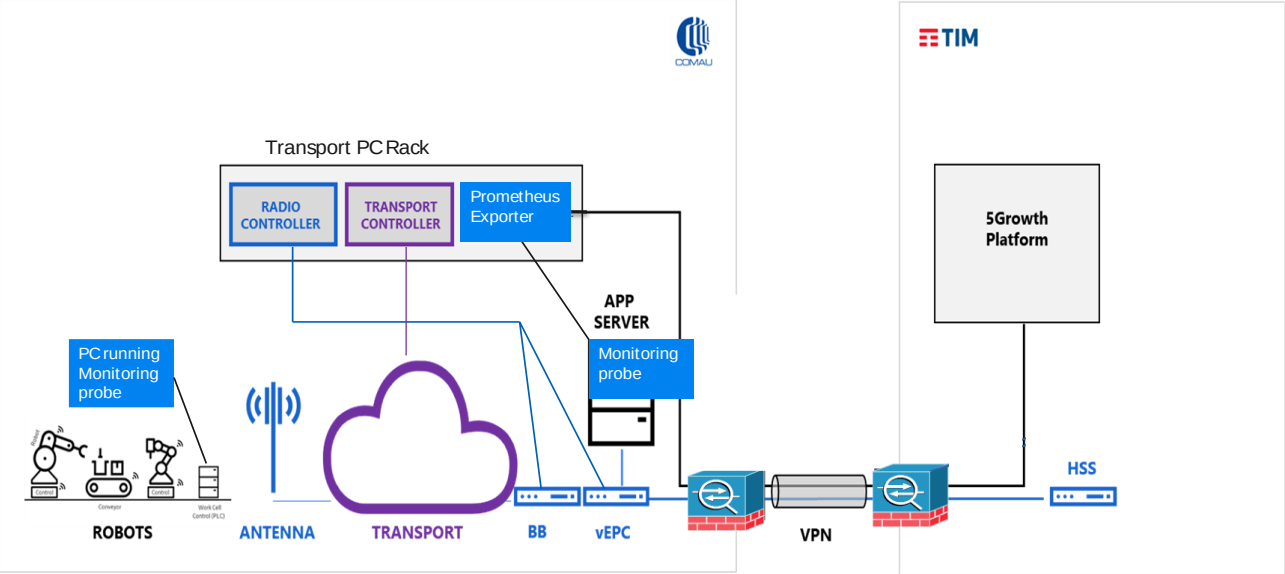

FIGURE 11: POSSIBLE 5GROWTH MONITORING PLATFORM PROBE DEPLOYMENT

3.2.4. Innovation Integration

It has been analyzed which of the innovations developed within WP2 could be integrated into the COMAU use cases. The following innovations are relevant for COMAU pilot: (i) Innovation I1 RAN Segment in Network Slice, (ii) Innovation I2 Vertical Service Monitoring, (iii) Innovation I3 Monitoring Orchestration.

- Innovation I1 is used for all use cases. It enables the creation of RAN slicing inside Pilot radio infrastructure allowing the handling traffic differentiation through the three use cases. To this end, each use case has a Network Slice Template (NST) that defines the QoS requirements of the service (e.g. service type or SST, bandwidth, delay). The NST is loaded in 5Gr-VS and the relative QoS parameters are used when the network slice is instantiated by 5Gr-SO and 5Gr-RL modules.

- Innovation I2 is used for Digital Twin use case (UC1). Two probes are installed on the endpoints of the service (one probe inside the robotic cell, the other inside the server where it runs the Digital Twin application) and the measures collected by the probes (that are the delay, jitter and packet loss) are sent to 5Gr-VoMS for aggregation and logging scope.
- Innovation I3 is also used for Digital Twin use Case (UC1). The monitoring system is triggered by 5Gr-SO when UC1 service is instantiated (using the network slice parameters defined for Innovation I1). The triggering will activate the probes to collect measures and report them to 5Gr-VoMS.

3.3. EFACEC_S Pilot

3.3.1. Technical Requirements and Related KPIs

The project proposal, in Table 1-6, anticipated the expected technical requirements for the EFACEC_S Pilot. These requirements have been further detailed in D1.1. Service KPIs, associated with the two use cases of the EFACEC_S Transportation pilot, have finally been mapped in Core KPIs which are the entities verified by measurements or qualitative considerations. In D4.1 the relevant Service KPIs for the two Use Cases were described and the association between Service KPIs and Core KPIs are reported in Tables 20 and 21 of D4.2.

Concerning the validation campaigns the following strategy was performed:

1. Validation of Transportation Pilot Use Cases over a real 5G network (5G SA network integrating ASOCS RAN and Fraunhofer Open5GCore) at IT Aveiro labs (IT2) and
2. Validation of the Transportation Pilot Use Cases over a real 5G network at vertical premises (Aveiro harbour)

The first validation (described in a) environment intends to verify the performance improvements in the 5G network related to the previous validation campaign. This validation allows also to have a baseline information to compare the measurements with the results obtained in the real vertical environment.

The second validation (described in b) environment will be deployed in Aveiro Harbour, for the second and for future validation campaigns that will be performed whenever relevant improvements in the 5G network will occur, both at performance level as at functional level.

The necessary considerations concerning the Core KPIs assessed in the tests are detailed below.

- **Use Case 1 (5GR-SKPI-2)** "Sync between communication components", which maps to the core CKPI-1 "e2e latency" and CKPI-2 "packet loss": The e2e latency is measured between both extreme equipment (5G-CPE related to rail sensor/strike IN detector to 5G-CPE related to Level Crossing Controller). Using the previous ping tool, the obtained value is the RTT and the latency value is calculated by the formula RTT/2, assuming that the UL (uplink) and the DL (down link) times are symmetrical. When the 5G monitoring platform is used to collect this measurement, the obtained result by the probes, running in both extreme equipment, is

the E2E latency. The packet loss is also measured between both extreme equipment (5G-CPE related to rail sensor/strike IN detector to 5G-CPE related to Level Crossing Controller). Using the previous Iperf3 tool, it is possible to get the packet loss for several bit rates under test. For this measurement, since the results obtained by the probes, related to 5G monitoring platform, are related to low bitrates, the *iperf* values are more relevant for this purpose.

- **Use Case 1 (5GR-SKPI-10)** "Service availability", which maps to the core CKPI-5 "Availability": According with the Service KPI definition the measurement is related to the percentage of time the service (5G network) is available to assure the proper behaviour of the level crossing without failures. For this Use Case, it is mandatory to conduct an evaluation time period at least of one month of uninterrupted 5G network service (at vertical premises) to calculate the Service availability.
- **Use Case 2 (5GR-SKPI-2)** "Sync between communication components", which maps to the core CKPI-1 "e2e latency" and CKPI-2 "packet loss": The e2e-latency is measured between both extreme equipment (Train driver console to the video Camara related to Level Crossing area). Using the previous ping tool, the obtained value is the RTT and the latency value is calculated by the formula RTT/2, assuming that the UL (uplink) and the DL (down link) times are symmetrical. When the 5G monitoring platform is used to collect this measurement, the obtained result by the probes, running in both extreme equipment is the E2E latency. The GUI running in the train driver console (onboard train) also reports the real time latency value, while the video is being transmitted. The packet loss is measured between both extreme equipment (i.e., train driver console to 5G-CPE related to the Level Crossing video camara). Using the previous Iperf3 tool, it is possible to get the packet loss for several bit rates under test. For this measurement, since the results obtained by the probes, related to 5G monitoring platform, are related to low bit-rates, the iperf values are more relevant for this purpose.
- **Use Case 2 (CKPI-3 "Guaranteed Data Rate"):** This measurement is achievement between both extreme equipment (train driver console to 5G-CPE related to the Level Crossing video camara). Using the previous Iperf3 tool, it is possible to get the guaranteed data rate. This measurement is also being collected by the 5G monitoring platform that shows the statistics of the tunnel interface established to transmit the video stream. The GUI, running in the train driver console (onboard train), also reports the guaranteed data rate value while the video is being transmitted. Due to the mobility requirements of the Use case 2, it is important to collect the data rates available to the video stream for different speeds of the vehicle. For this measurement, it is used the GUI running in the Driver console to register the data rate according to different speeds. For this specific purpose, a car simulating a train will be used due to the logistic restrictions regarding the train.
- **Use Case 2 (CKPI-9 "Jitter"):** This measurement is collected between both extreme equipment (train driver Console to 5G-CPE related to the Level Crossing video camara). Using the previous Iperf3 tool, it is possible to get the jitter values regarding different data rates. This measurement is also being collected by the monitoring platform. The GUI, running in the train driver console (onboard train) also reports the jitter value while the video is being transmitted.

3.3.2. Measurement Procedures

For this validation campaign the measurement procedures, already described in D4.2, section 2.2.3 will be used but we will also report the results obtained by the 5G monitoring platform. Additionally, it is important to mention that the Use Case 2 provides, at the application level (GUI), the information regarding the KPI measurements.

3.3.3. ICT-17 and 5Growth Platform Integration

The integration between the 5G-VINNI Monitoring platform and the 5Growth Monitoring Platform was done by the SDA described in section 2.1.1.

The 5G-VINNI Monitoring platform is part of the Sonata Framework and uses a Prometheus Server and a Prometheus Pushgateway. The data can be collected from the Prometheus server using the REST API. To this end, the SDA has a collector for this kind data source (Prometheus-based data source) described in section 2.1.2.

The 5Growth Monitoring Platform can be seen as a data consumer of SDA (see section 2).

The process to have the monitoring data is separated into two steps:

- First of all the, the pilots components need to have a configured probe to send periodically data metrics to the 5G-VINNI Platform (to the Prometheus Pushgateway, and then the Prometheus Server collects the data from Prometheus Pushgateway);
- Next, the SDA needs to be configured to collect this specific data metrics from 5G-VINNI Platform and provides them to the 5Growth Monitoring Platform.

When this process was done, the pilots can start the probes, and the metrics can be seen in the 5Growth Monitoring Platform.

3.3.4. Innovation Integration

This pilot will consider the integration of the I6 inter-domain and I11 security innovation. Regarding the former, the innovation will allow the vertical obtaining a connectivity service that is handled by the 5Growth Vertical Slicer existing in one domain, sending service creation requests to the network service provider in another domain. This service, composed by slices belonging in different domains, can be dynamically reconfigured considering different stimuli, such as increasing the security level of the link (i.e., by increasing the mutation period of the service ports from the MTD service, or requesting the channel to be built with a different bandwidth). As for the later, the Moving Target Defense mechanism will serve as the underlying security mechanism between the Vertical Slicer and the SONATA driver belonging to the 5G-VINNI deployment (where the network services and connectivity necessary to support the vertical services is made available). This joint operation of the I6 inter-domain and I11 security innovations provides the added capability of ensuring the network control interaction between remote peers, thus paving the way for a new kind of safety-certified classification, which is appreciated in the public utility sector. This integration of innovations will also be exploited in the EFACEC_S pilot besides the EFACEC_E one. However, in the EFACEC_E case, the

two domains will be deployed in different providers physically deployed in the Campus, whereas in the EFACEC_S these will be deployed between the Campus and the Aveiro harbour.

3.4. EFACEC_E Pilot

3.4.1. Technical Requirements and Related KPIs

The 5G technical requirements concerning the two use cases of the Energy pilot, UC1 & UC2, were initially anticipated in the 5GROWTH Proposal, in Table 1-6.

Along the project execution, the 5G technical requirements were further detailed in D.1.1, Tables 24 and 27. In D4.1 the Service KPIs for the two Energy use cases were described, and in D4.2 the relationship between these Service KPIs and the Core KPIs was established and reported in Tables 22 and 23.

The Core KPIs are measured on the vertical premises. However, the UC1 setup is not the final one since the secondary substation automation is still deployed in the IT lab. The measurements are done on the 5G SA network integrating ASOCS RAN and Fraunhofer Open5GCore. Section 2.2.4 of D4.2 reports the setup used for performing measurements on the 5G network.

The necessary considerations concerning the Core KPIs assessed in the tests are detailed below.

- **CKPI-1 End-to-end Latency** - Regarding end-to-end latency, the two use cases present different realities: UC1 end-to-end latency is composed mainly by the latency in two segments: The 5G segment (5G CPE – 5G RU – 5G Core), and the wired segment (5G Core – Datacenter VMs). It is thus possible to decompose the end-to-end latency into two contributions and evaluate the impact of each segment. After having relied on the ICMP ping for the preliminary campaigns to measure the latency, the probe developed by SSSA "End-to-end Unidirectional Link Latency Evaluator" (in D4.2 section 2.3.1) has been implemented. Using this probe across both UC1 and UC2 KPI measurement setup allows to automatically collect the measurement data and feed it to the Prometheus monitoring platform. One of the probes is installed in the Frontend VM and the other one is installed on a raspberry-pi device connected between the GSmart and the 5G CPE, since it was technically not possible to install the probe directly in the GSmart.
  Concerning the UC2, the user end devices – Gsmart and LVS3 - are both connected to 5G CPEs, with a raspberry-pi device in the middle. Again, this was the solution adopted due to not being technically possible to install the probes directly in the LVS3 and the Gsmart devices
- **CKPI-2 Packet Loss** - The Packet Loss KPI is measured using the SSSA probes in both use cases UC1 and UC2 and the collected data is integrated with the Prometheus monitoring platform.
  The packet loss is also measured using iPerf3 for UDP traffic, and the Tx retries are measured using iPerf3 for TCP traffic, as this is a relevant indicator when assessing the quality of the telemetry transfer between the Secondary Substation and the Control Center.

- **CKPI-3 Guaranteed Data Rate** - The Guaranteed Data Rate KPI is measured in UC1 using iPerf3 in UL, for the communication between the Secondary Substation and the Control Center, and in UL and DL, for the communication between the Secondary Substation and the mobile device belonging to the field maintenance crew.
  Date rate measurements are made for both TCP and UDP traffic.
  Additionally, the throughput of real application traffic is measured between the Secondary Substation and the Control Center, using the IT probe. Applying this probe in use cases UC1 setup allows to integrate with the Prometheus monitoring platform.
- **CKPI-5 Availability** - The Availability KPI refers to the percentage of time during which the radio network ensures the support of the services with the expected QoS. It is the ratio between the up time of the radio link over the total time of the radio link service, considering the total time as being a given observation period.
  Considering the nature of the project and the fact that the 5G network solution used in the Energy pilot is a non-commercial product, it does not make sense trying to measure the network availability. Conversely, it is measured the service availability considering the number of lost packets per total amount of transmitted packages during a transmission period.
- **CKPI-9 Jitter** - The Jitter KPI refers to the variation of latency between packets. The average jitter over a period is measured using the SSSA probes in both use cases UC1 and UC2 and the collected data is integrated with the Prometheus monitoring platform.

3.4.2. Measurement Procedures

This information is already detailed in D4.2, Section 2.2.4.

3.4.3. ICT-17 and 5Growth Platform Integration

The integration between the 5G-VINNI Monitoring platform and the 5Growth Monitoring Platform was done by the component SDA described in section 2.1.1.

The 5G-VINNI Monitoring platform is part of the Sonata Framework and uses a Prometheus Server and a Prometheus Pushgateway. The data can be collected from the Prometheus server using the REST API. To this end, the SDA has a collector for this kind data source (Prometheus-based data source). This this collector is described in section 2.1.2.

The 5Growth Monitoring Platform can be seen as a data consumer of SDA, and SDA developed this process, described in section 2 can be separated into two steps:

- First of all, the, the pilots' components need to have a probe configured to send periodically data metrics to the 5G-VINNI Platform (send to Prometheus Pushgateway, and then Prometheus Server collects from Prometheus Pushgateway).
- Next, the SDA needs to be configured to collect this specific data metrics from 5G-VINNI Platform and provide them to the 5Growth Monitoring Platform.

When this process was done, the pilots can start the probes, and the metrics can be seen in the 5Growth Monitoring Platform.

3.4.4. Innovation Integration

This pilot will consider the integration of the I6 inter-domain and I11 security innovation. Regarding the former, the innovation will allow the vertical to obtain a connectivity service that is handled by the 5GROWTH Vertical Slicer existing in one domain, sending service creation requests to the network service provider in another domain. This service, composed by slices belonging in different domains, can be dynamically reconfigured considering different stimuli, such as increasing the security level of the link (i.e., by increasing the mutation period of the service ports from the MTD service, or requesting the channel to be built with a different bandwidth). As for the later, the Moving Target Defense mechanism will serve as the underlying security mechanism between the Vertical Slicer and the SONATA driver belonging to the 5G-VINNI deployment (where the network services and connectivity necessary to support the vertical services is made available). This joint operation of the I6 inter-domain and I11 security innovations provides the added capability of ensuring the network control interaction between remote peers, thus paving the way for a new kind of safety-certified classification, which is appreciated in the public utility sector. This integration of innovations will also be exploited in the EFACEC_S pilot besides the EFACEC_E one. However, in the EFACEC_E case, the two domains will be deployed in different providers physically deployed in the Campus, whereas in the EFACEC_S these will be deployed between the Campus and the Aveiro Harbour.

4. Report of the Second Validation Campaign

Within 5Growth, pilot use cases are validated in the context of the successive releases of the experimental environments and reported in the corresponding deliverable. Once we have described the methodology for these experiments, the characteristics of the KPIs, their integration with ICT-17 and the 5Growth platform, and the innovations introduced by this project, this section focuses on the actual report for the second validation results. The results from first validation campaign can be found in D4.2 section 5.

4.1. Industry 4.0 Pilot – INNOVALIA

This section includes the results obtained from the tests performed around the use cases within the INNOVALIA pilot.

4.1.1. Use Case 1: Connected Worker Remote Operation of Quality Equipment

The final setup implemented at the 5TONIC lab was the following:

- The devices on each end were connected to a different 5G Customer Premises Equipment (5G CPE).
- Both 5G CPEs were connected to the same cell. This means that radio resources are shared if both CPEs are transmitting at the same time.
- The video device used is a raspberry pi camera.
- The scanner is Innovalia's CMM product.
- The software to operate the scanner is called M3, and the application has some functionalities implemented in the VM deployed on the Data Center, called INNO VM in Figure 12.

Following picture (Figure 12) shows the equipment connected to the 5G network in the 5TONIC lab and the IP assignment for the interconnectivity.

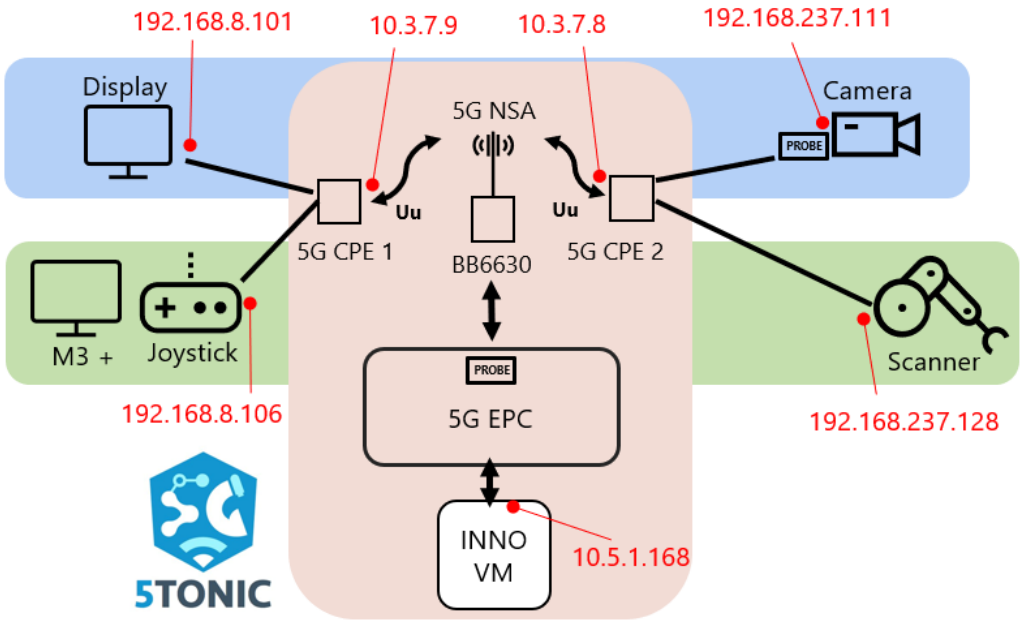

FIGURE 12: INNOVALIA UC1 SETUP

An operator sent commands to the CMM through the M3 software that told the CMM arm how to move to measure the piece placed in the scan area. In return, the CMM sent the results of the optical scan towards the M3 end. The performance of this traffic was measured by a 5Probe installed at the entry router of the 5G network core.

At the same time, all the movements of the CMM arm were recorded by the camera and streamed over the same 5G network. The RTT latency and jitter of this traffic was measured by a 5Probe installed at the raspberry pi. Several iterations of this experiment have been performed, all with similar results. Some graphs of one of the iterations are shown below.

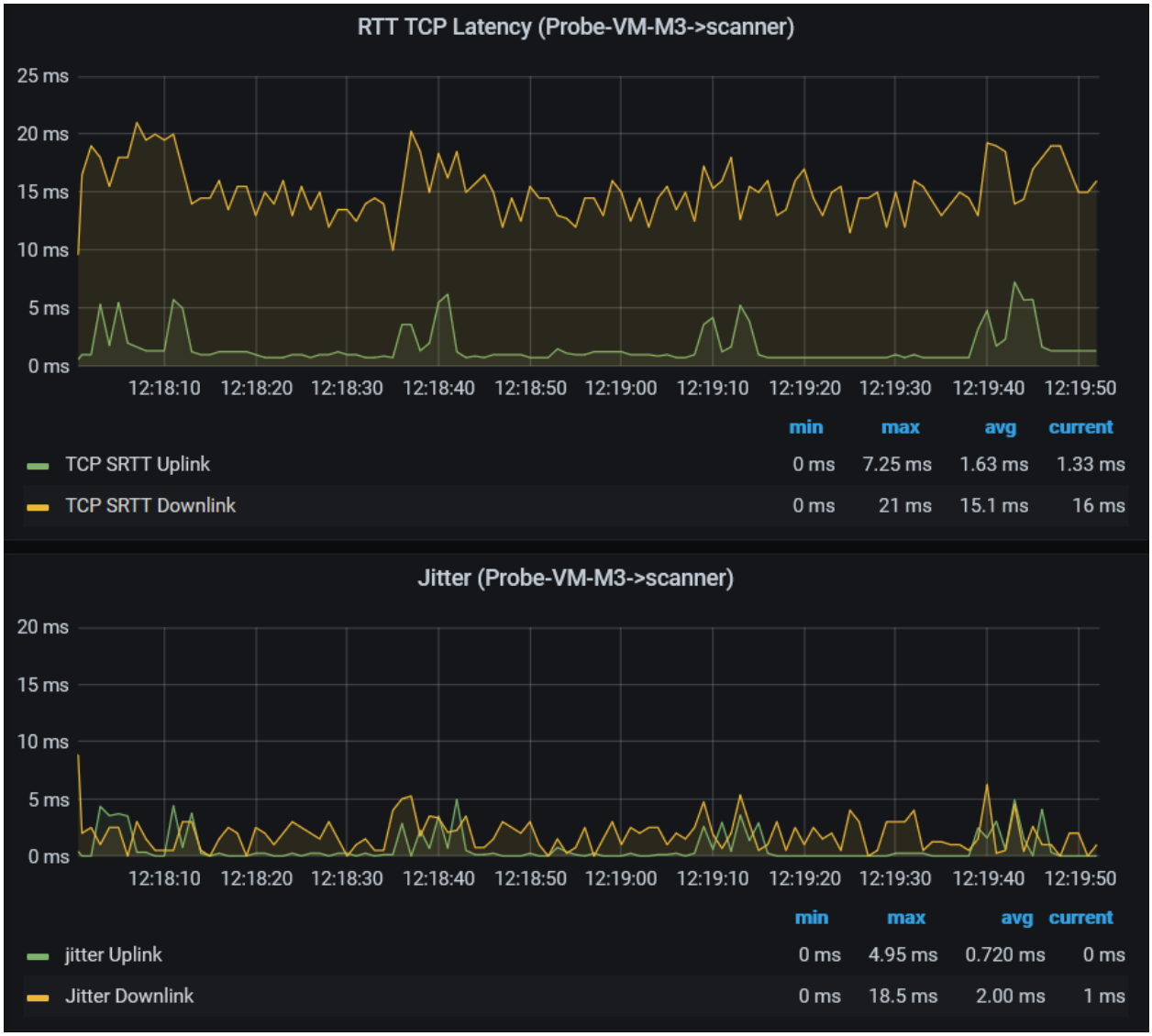

FIGURE 13: RTT LATENCY AND JITTER FOR TRAFFIC FROM M3 TO SCANNER

The downlink line shows the RTT TCP latency and jitter measured between VM and CMM (commands traffic). It is around 14-16 ms latency when there is traffic.

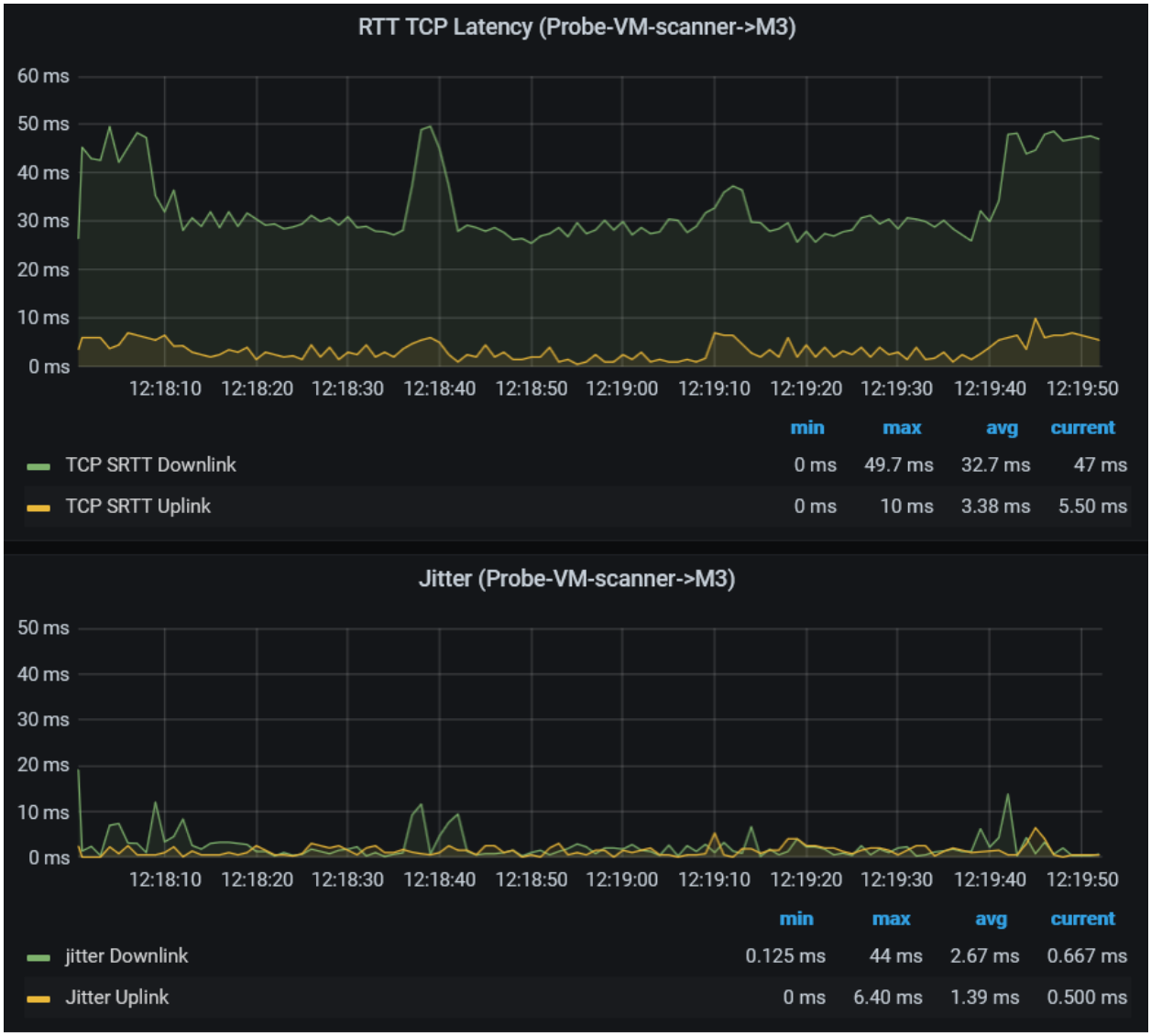

FIGURE 14: RTT LATENCY AND JITTER FOR TRAFFIC FROM SCANNER TO M3

The uplink line shows the RTT TCP latency and jitter measured between the VM and the M3 (scan results traffic). It is around 28-30 ms latency when there is traffic.

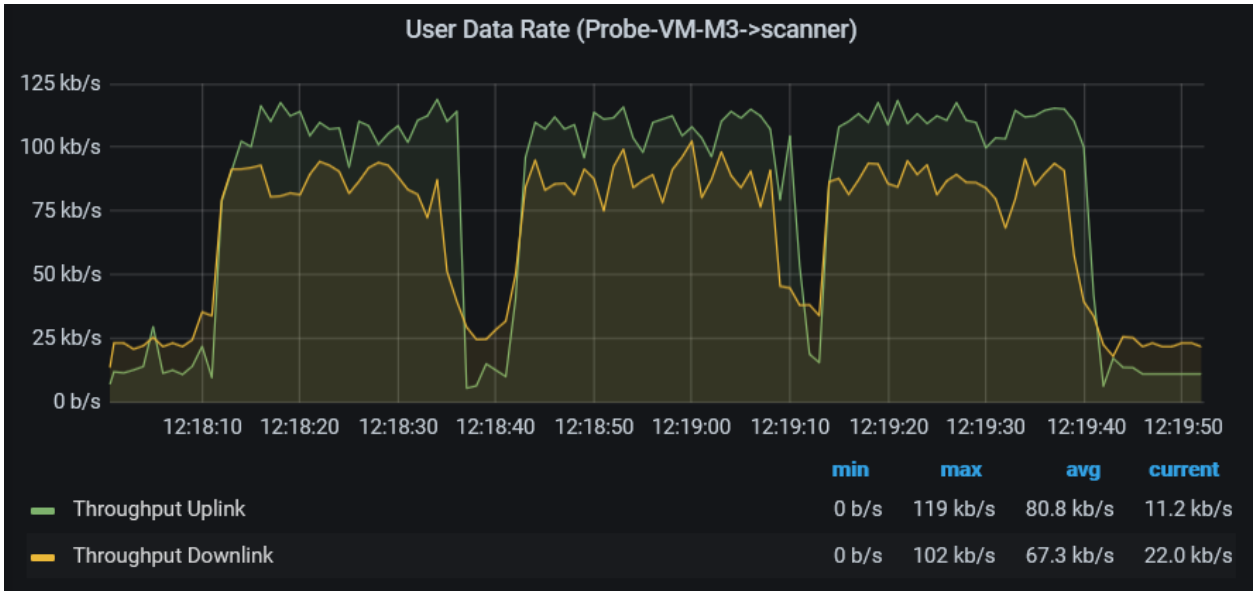

FIGURE 15: USER DATA RATE FROM M3 TO SCANNER

Figure 15 shows the User Data Rate measured at the VM for the traffic from M3 towards CMM (commands traffic). Max of 130 Kbps for traffic from M3 to VM. Around 100 Kbps measured for traffic from VM towards CMM.

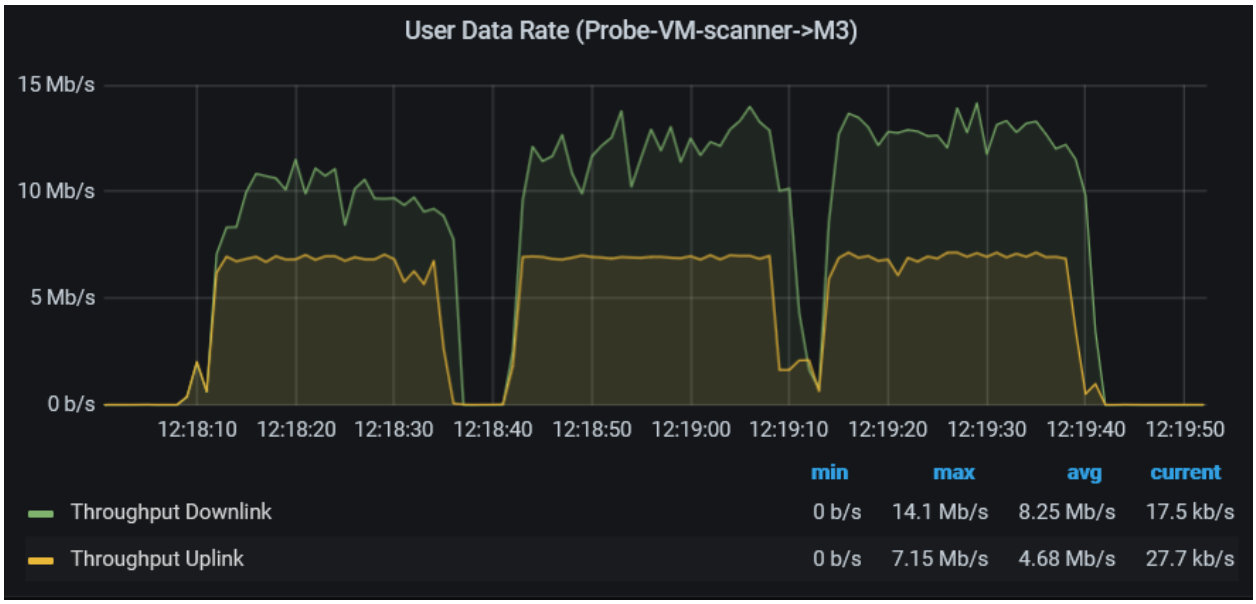

FIGURE 16: USER DATA RATE FROM SCANNER TO M3

Figure 16 shows the User Data Rate measured at the VM for the traffic from CMM towards M3 (scan results traffic). Traffic going out the VM towards the M3 demands and reaches around 13 Mbps (downlink). Traffic coming from CMM to VM demands and reaches around 7 Mbps (uplink).

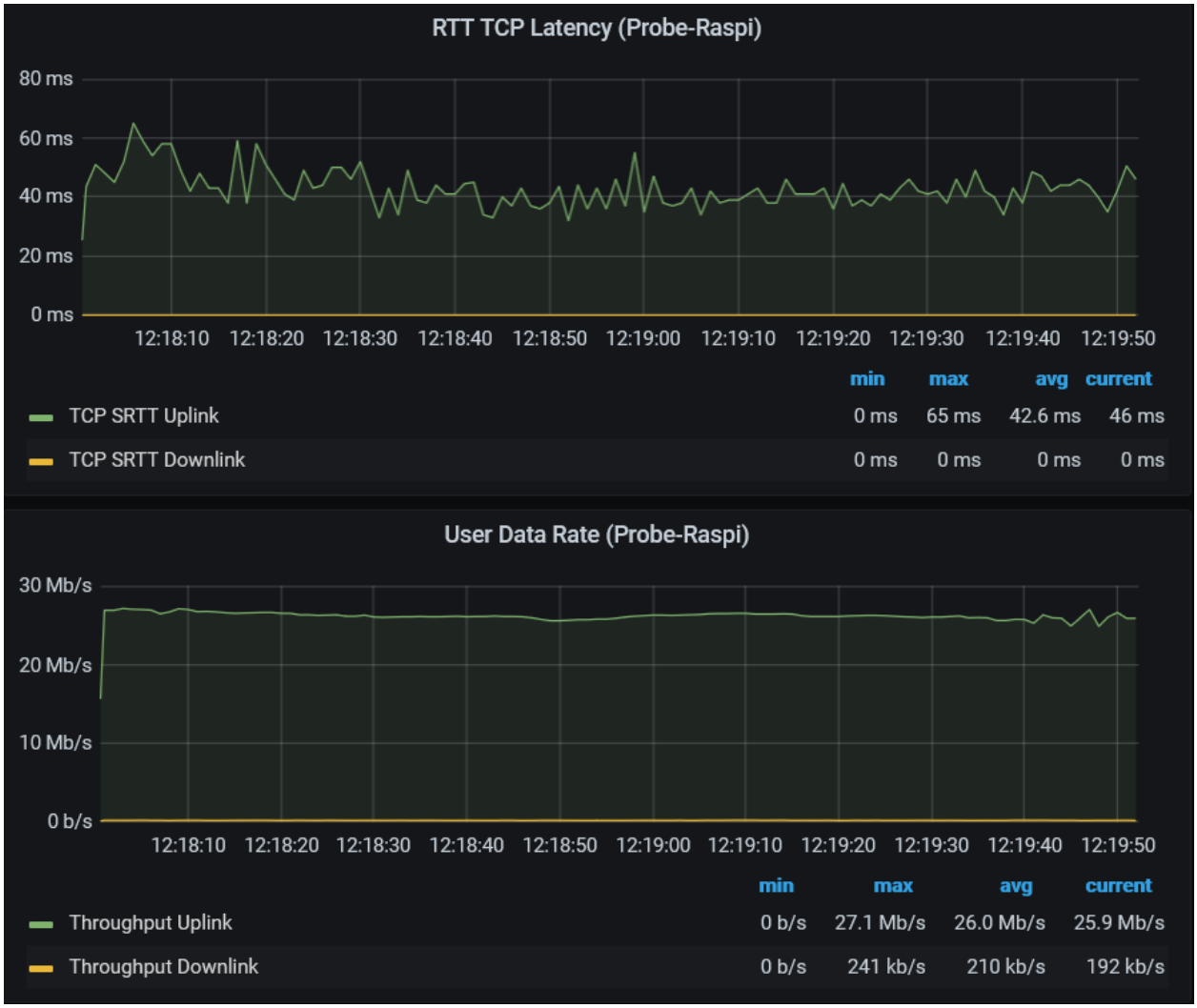

FIGURE 17: VIDEO STREAMING LATENCY AND DATA RATE

Figure 17 shows the RTT TCP Latency of the video streaming measured in the raspberry pi, averging45 ms. User Data Rate of the video streaming is stable at 25 Mbps.

The results obtained in this first experiment are summarized in Table 7.

TABLE 7: INNOVALIA UC1 MEASURED METRICS

SKPI	CKPI	CKPI result	SKPI result
P1UC1-SKPI-1: Teleworker-CMM Synchronization (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	M3: 30 ms + 15 ms Video: 45 ms	Good.
	CKPI-2 Packet Loss	0	
P1UC1-SKPI-2: High-resolution Real-time Video Quality (=5GR-SKPI-4)	CKPI-2 Packet Loss	0	4 out of 5.
	CKPI-3 Guaranteed Data Rate	Video: 25 Mbps	
	CKPI-9 Jitter	90% under 3 ms	

P1UC1-SKPI-4: Radius of Operation (=5GR-SKPI-5)	CKPI-1 End-to-end Latency	M3: 30 ms + 15 ms Video: 45 ms	0 km.
	CKPI-5 Availability	-	

Further testing was done under the same setup conditions as detailed above to obtain the maximum distance between both ends that would still allow an acceptable performance of the use case. For that purpose, extra latency was added on the M3 host in steps of 10 ms, to find the maximum latency value and then translate that value to distance, assuming that every 100 km traversed by a packet adds half a millisecond to the one-way latency, i.e., 1 ms to the RTT [20].

It was found that adding up to 30 ms of extra latency on the M3 host, the synchronization level was acceptable for the remote operator, but the operation experience was ranked as bad when setting 40 ms. See below the graphs obtained for those tests.

With an extra 30 ms, the measured metrics were the following:

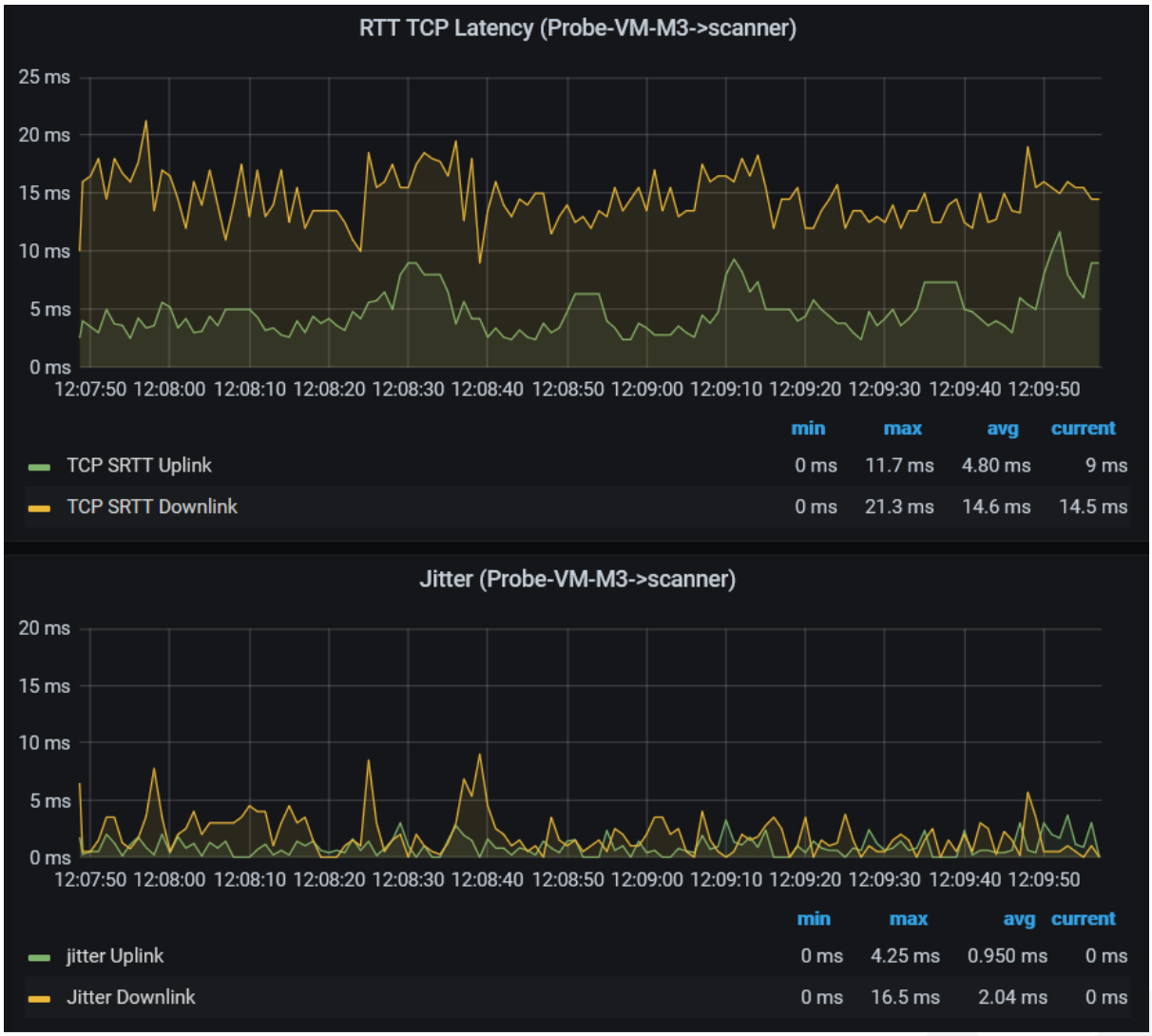

FIGURE 18: RTT LATENCY AND JITTER FROM VM-M3 TO SCANNER

Figure 18 shows the RTT TCP Latency and jitter measured between VM and CMM (commands traffic). It is around 14-16 ms latency when there is traffic. In this segment, between VM and CMM, there is no impact from the extra latency.

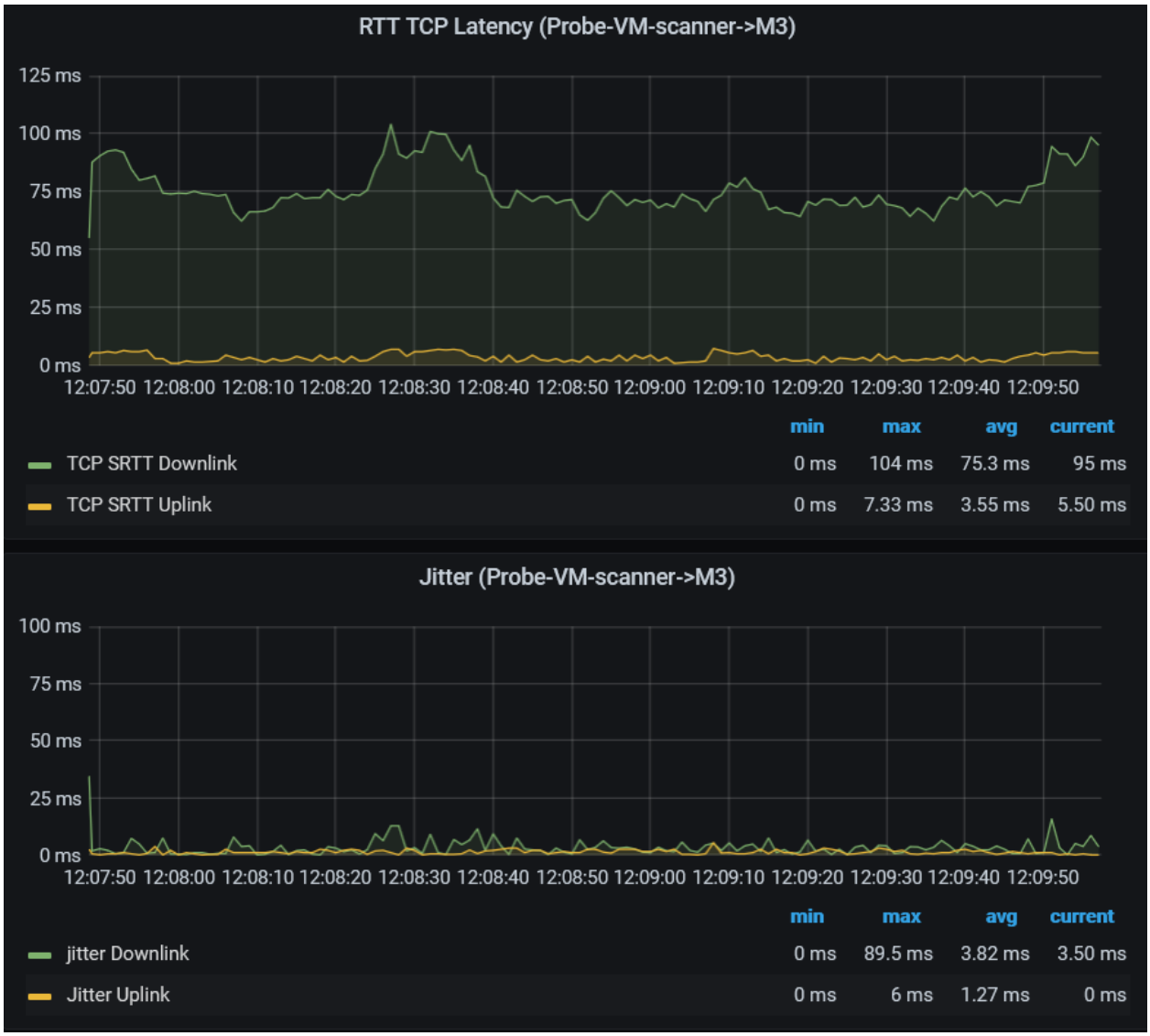

FIGURE 19: RTT LATENCY AND JITTER FROM VM-SCANNER TO M3

Figure 19 shows the RTT TCP latency and jitter measured between the VM and the M3 (scan results traffic). It is around 70 ms latency when there is traffic. As expected, the metrics are showing the impact of the extra delay.

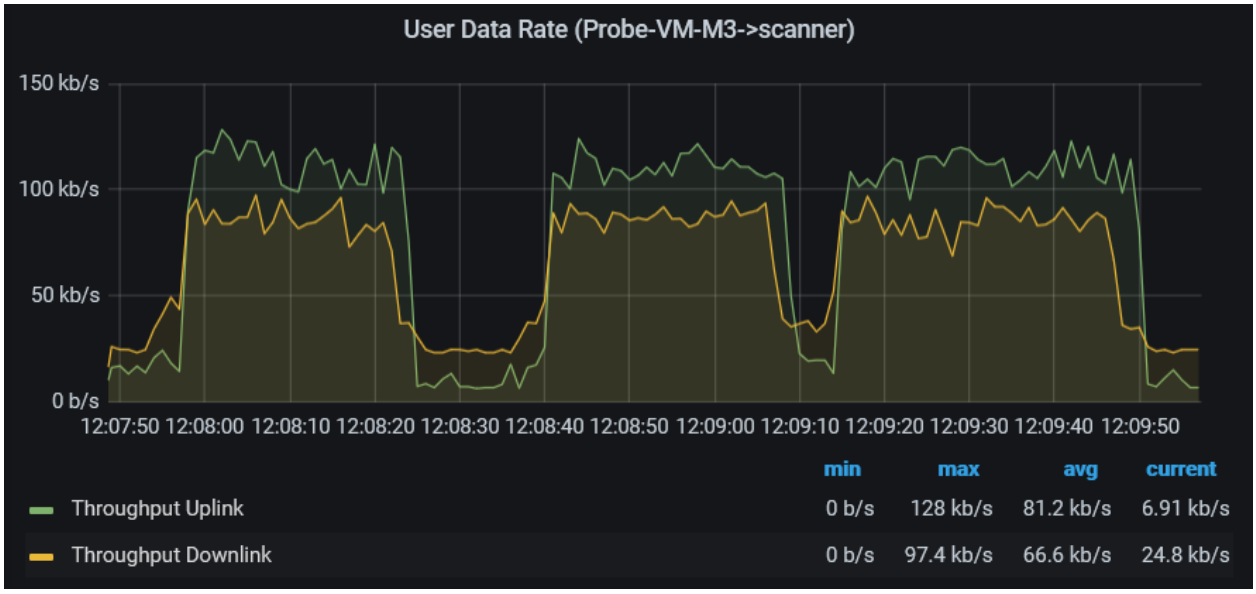

FIGURE 20: USER DATA RATE VM-M3 TO SCANNER

Figure 20 shows User Data Rate is measured at the VM for the traffic from M3 towards CMM (commands traffic). The Max of 130 Kbps for traffic from M3 to VM is achieved. Around 100 Kbps is measured for the traffic from VM towards CMM. No noticeable change.

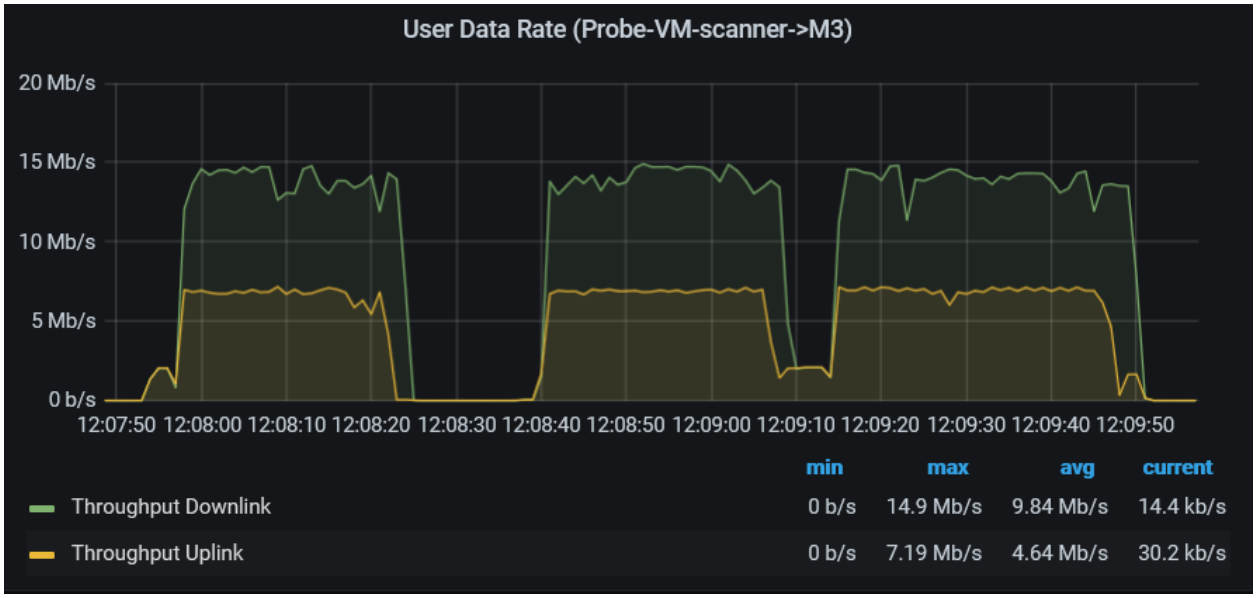

FIGURE 21: USER DATA RATE VM-SCANNER TO M3

Figure 21 shows User Data Rate is measured at the VM for the traffic from CMM towards M3 (scan results traffic). Traffic going out the VM towards the M3 reaches around 13 Mbps. Traffic coming from CMM to VM reaches around 7 Mbps. No noticeable change.

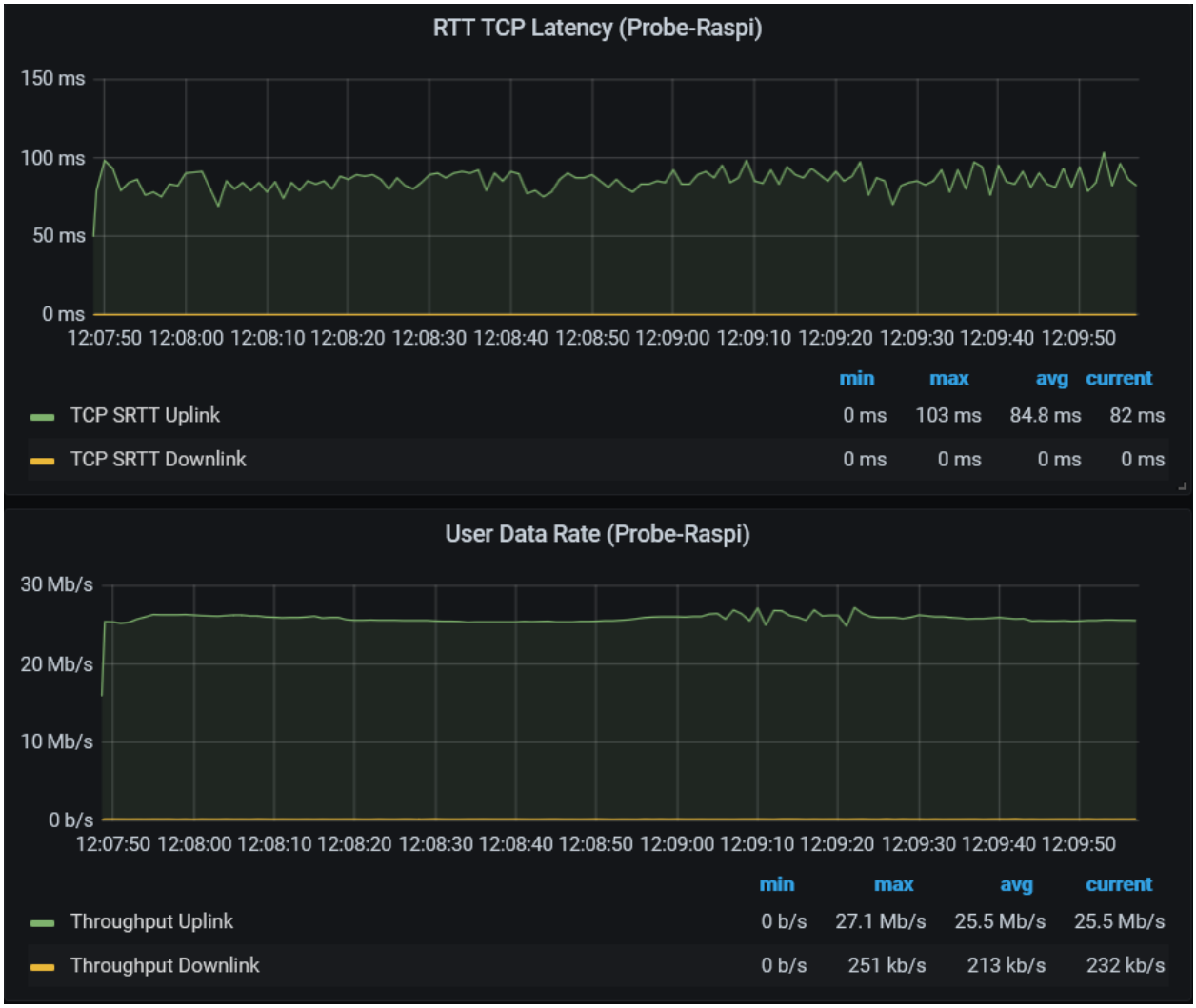

FIGURE 22: RTT LATENCY AND USER DATA RATE OF THE VIDEO STREAMING

Figure 22 shows the RTT TCP Latency of the video streaming is measured in the raspberry pi is around 85 ms. Data Rate of the video streaming is a constant 25 Mbps.

TABLE 8: KPIS WITH ADDED 30 MS

SKPI	CKPI	CKPI result	SKPI result
P1UC1-SKPI-1: Teleworker-CMM Synchronization (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	M3: 70 ms + 15 ms Video: 85 ms	Acceptable.
	CKPI-2 Packet Loss	0	
P1UC1-SKPI-2: High-resolution Real-time Video Quality (=5GR-SKPI-4)	CKPI-2 Packet Loss	0	4 out of 5.
	CKPI-3 Guaranteed Data Rate	Video: 25 Mbps	

	CKPI-9 Jitter	90% under 3 ms	
P1UC1-SKPI-4: Radius of Operation (=5GR-SKPI-5)	CKPI-1 End-to-end Latency	M3: 70 ms + 15 ms Video: 85 ms	3000 km. The extra 30 ms added mean both ends would be 3000 km apart.
	CKPI-5 Availability	-	

With an extra 40 ms, the measured metrics were the following:

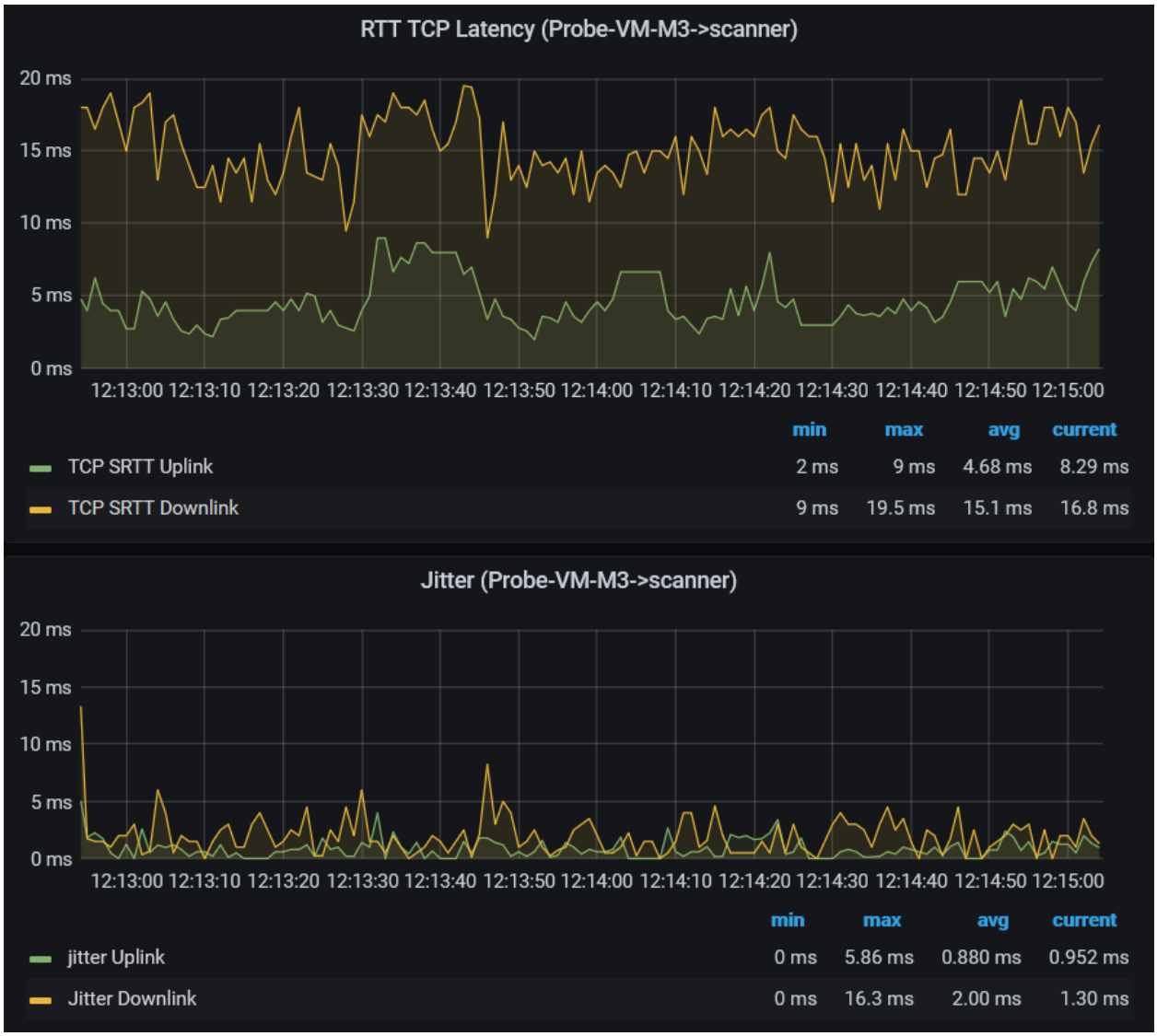

FIGURE 23: RTT LATENCY AND JITTER FROM VM-M3 TO SCANNER

Figure 23 shows the RTT TCP Latency and jitter measured between VM and CMM (commands traffic). It is around 16 ms latency when there is traffic.

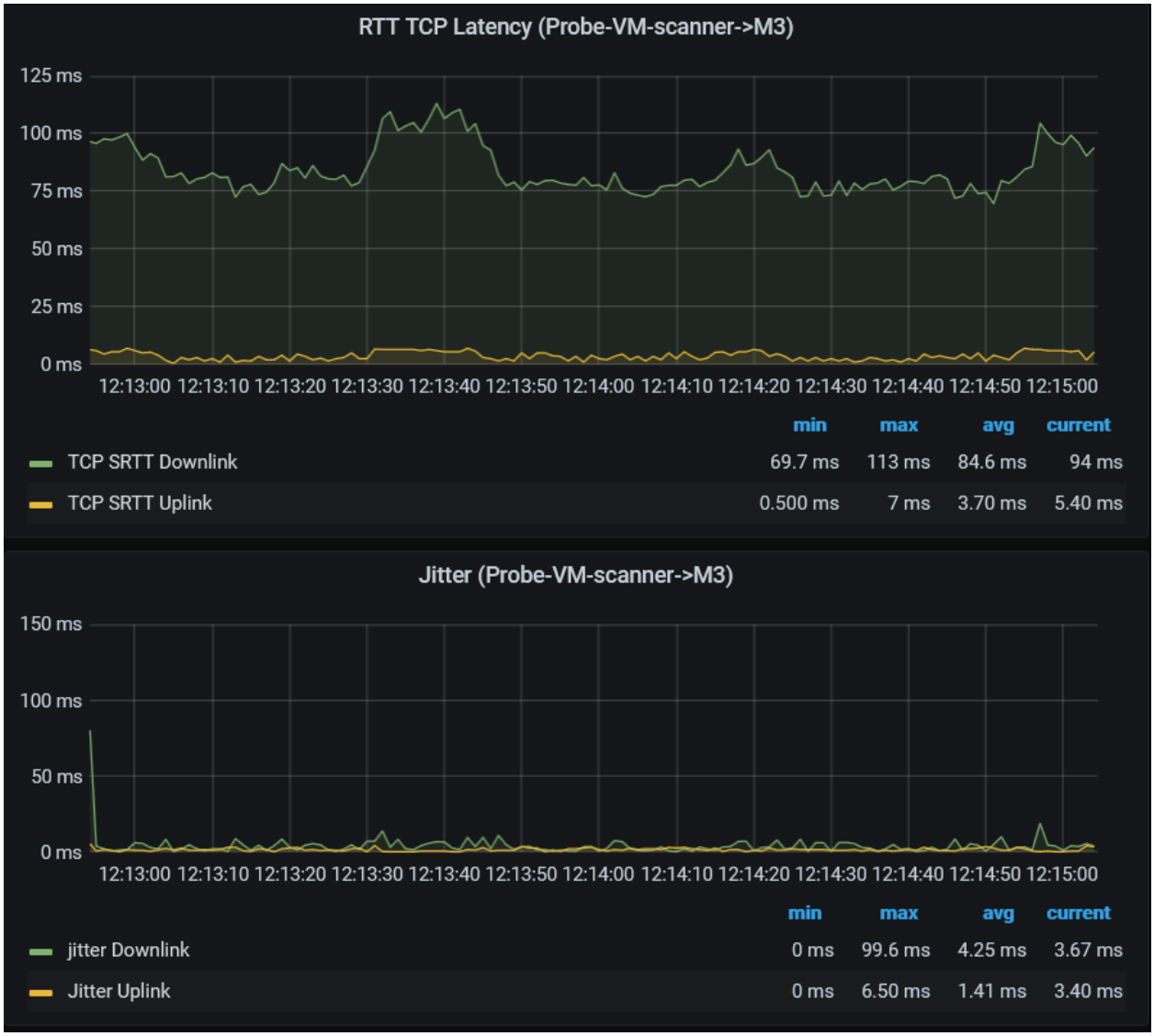

FIGURE 24: RTT LATENCY AND JITTER FROM VM-SCANNER TO M3

The uplink line in Figure 24 shows the RTT TCP latency and jitter measured between the VM and the M3 (scan results traffic). It is around 80 ms latency when there is traffic. As expected, the metrics are showing the impact of the extra delay.

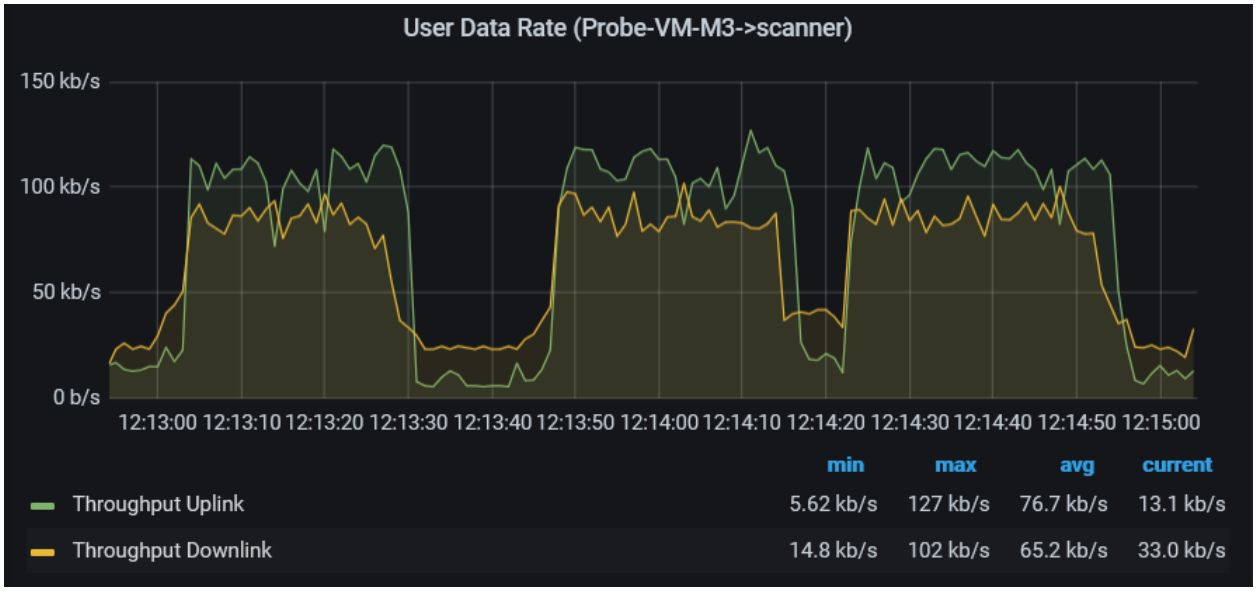

FIGURE 25: USER DATA RATE VM-M3 TO SCANNER

Figure 25 shows User Data Rate measured at the VM for the traffic from M3 towards CMM (commands traffic). Max of 130 Kbps for traffic from M3 to VM. Around 100 Kbps measured for traffic from VM towards CMM. No noticeable change.

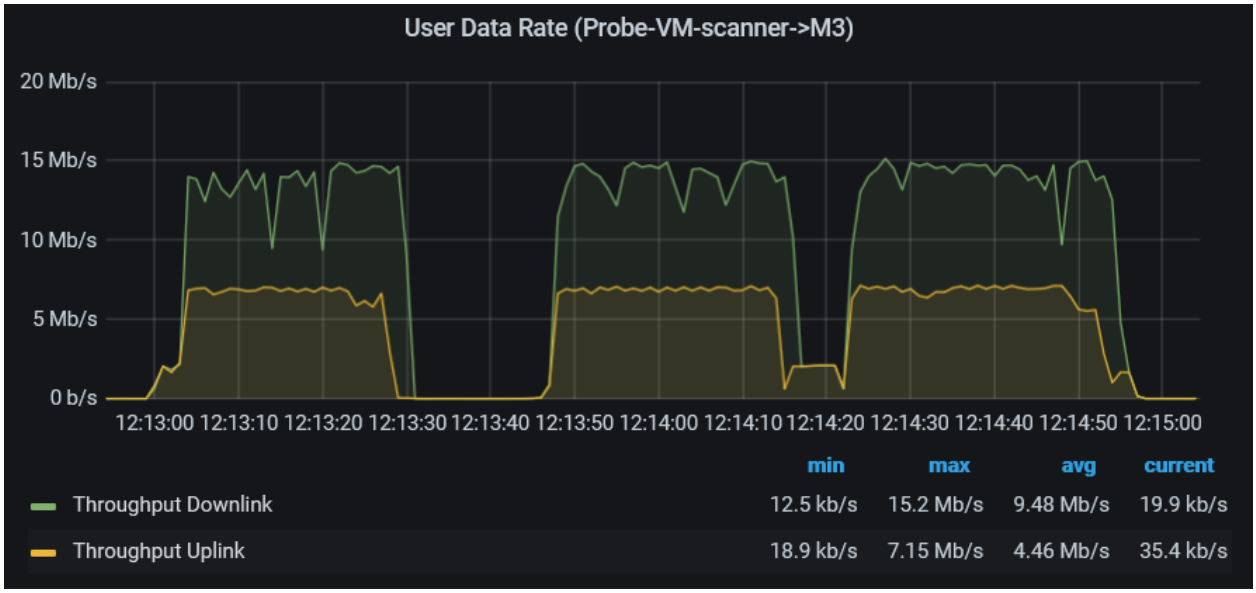

FIGURE 26: USER DATA RATE VM-SCANNER TO M3

Figure 26 shows User Data Rate measured at the VM for the traffic from CMM towards M3 (scan results traffic). Traffic going out the VM towards the M3 reaches around 13 Mbps. Traffic coming from CMM to VM reaches around 7 Mbps. No noticeable change.

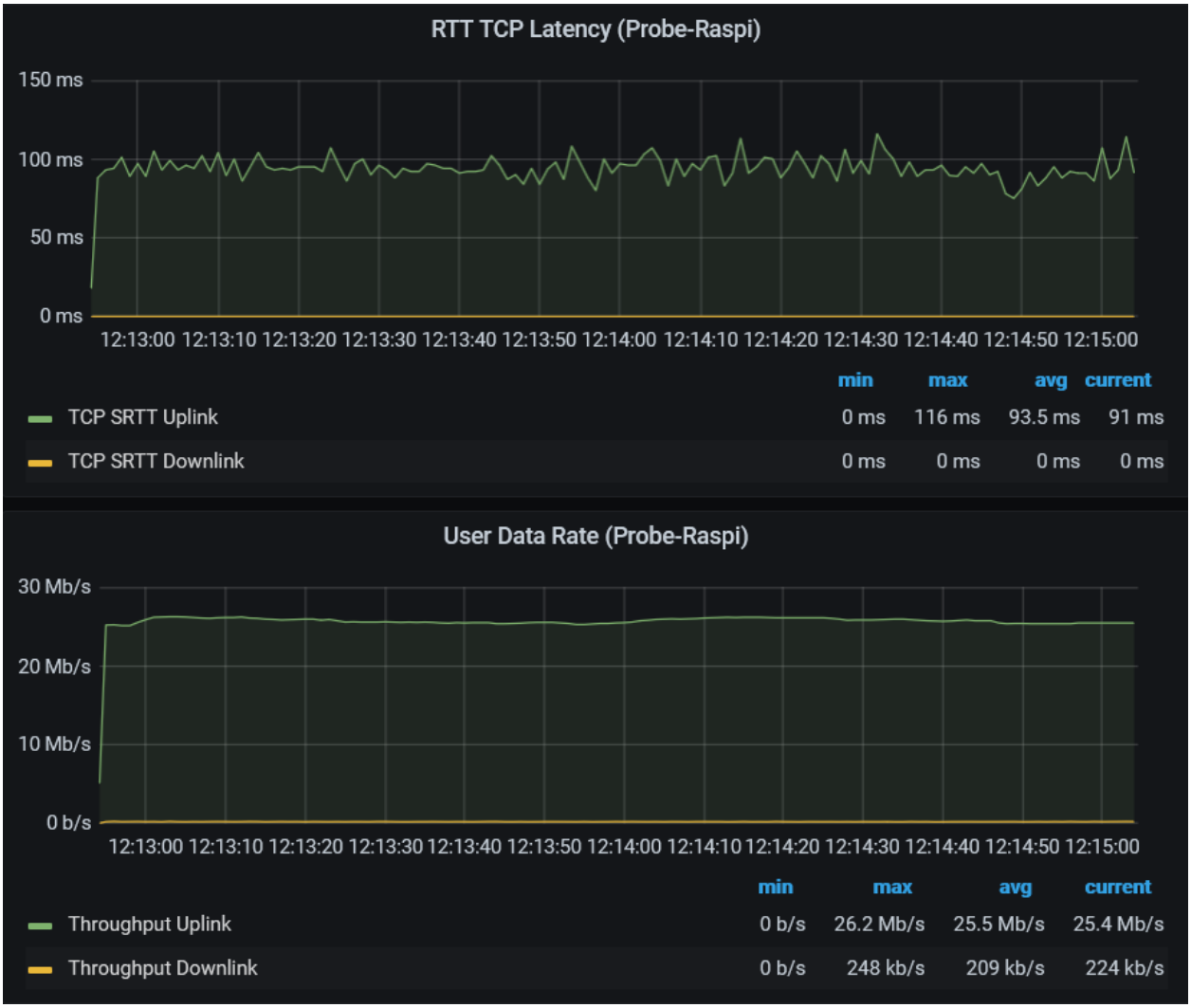

FIGURE 27: RTT LATENCY AND USER DATA RATE OF THE VIDEO STREAMING

Figure 27 shows the RTT TCP Latency of the video streaming measured in the raspberry pi is around 95 ms. Data Rate of the video streaming is a constant 25 Mbps.

TABLE 9: KPIS WITH ADDED 40 MS

SKPI	CKPI	CKPI result	SKPI result
P1UC1-SKPI-1: Teleworker-CMM Synchronization (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	M3: 80 ms Video: 95 ms	Bad.
	CKPI-2 Packet Loss	0	
P1UC1-SKPI-2: High-resolution Real-time Video Quality (=5GR-SKPI-4)	CKPI-2 Packet Loss	0	4 out of 5.
	CKPI-3 Guaranteed Data Rate	Video: 25 Mbps	

	CKPI-9 Jitter	90% under 3 ms	
P1UC1-SKPI-4: Radius of Operation (=5GR-SKPI-5)	CKPI-1 End-to-end Latency	M3: 80 ms Video: 95 ms	4000 km. The extra 40 ms added mean both ends would be 4000 km apart.
	CKPI-5 Availability	-	

In terms of vertical service setup time, Figure 28 presents the experimental results regarding the instantiation and termination times of the INNO-UC1 vertical service. These experimental results were measured in the 5TONIC testbed, where both 5Growth and 5G-EVE platforms are deployed on its Edge Datacenter. Each platform is managing their own set of computational and networking resources. Management, control, and data-plane connectivity between platforms is established through the 5TONIC network, without additional security mechanisms for cross domain connections. Such a setup aims to demonstrate the deployment of INNO-UC1 vertical service in a Public Network Integration with Non-Public Network (PNI-NPN), where the PN and the NPN domains are embodied by 5G EVE and 5Growth platforms, respectively.

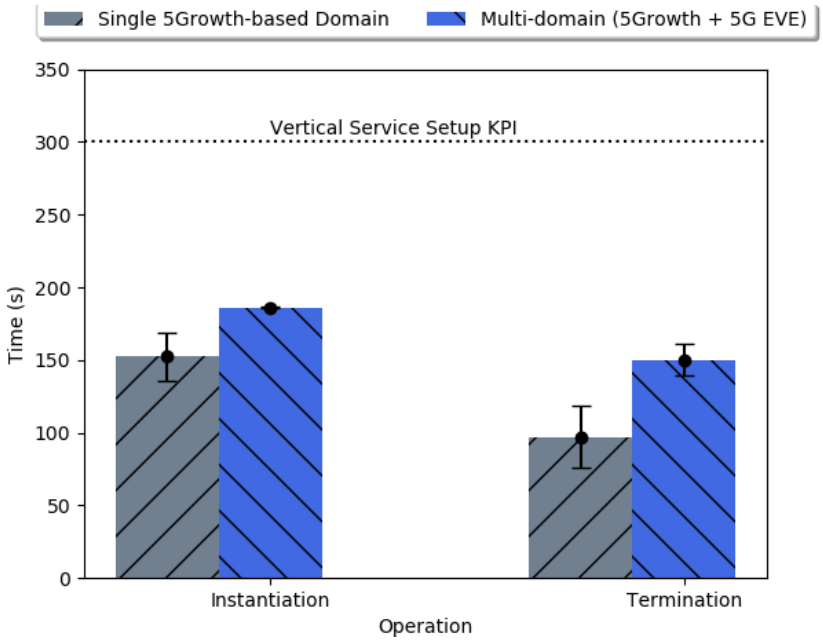

FIGURE 28: VERTICAL SERVICE SETUP FOR INNO-UC1

The total time of the instantiation and termination operations is about 186 s and 153 s, respectively, which is under the Vertical Service Setup Time KPI of 300s defined in Section 3.1.1. A more detailed time profiling of the whole process shows that: the most time-consuming operation is related to the creation and instantiation of the VNFs and the creation of the virtual links and networks, along with the termination and clean-up of the previous aspects. This step represents about 63% and 40% of the total time for, respectively, instantiation and termination operations.

When comparing to a deployment over a single 5Growth-based domain (i.e., not using 5G EVE), this represents an average increase of 34s and 53s for both the instantiation and termination operations.

The main reason for such an increase is the fact that both 5Growth and 5G EVE components follow and implement polling mechanisms (instead of an event-based approach) to verify the status of operations on their counterparts.

4.1.2. Use Case 2: Connected worker: Augmented Zero Defect Manufacturing (ZDM) Decision Support System (DSS)

Use case 2 adds to the use case 1 setup the following components:

- An Automated Guided Vehicle (AGV) that brings the pieces to the scan area placed in the same room as the CMM.
- The AGV is connected to another 5G CPE.
- A new AGV controller (AGVc) VM, which controls the movement of the AGV, deployed on the Data Center.

The Figure 29 shows the implementation at 5TONIC lab and the IP assignment for the interconnectivity.

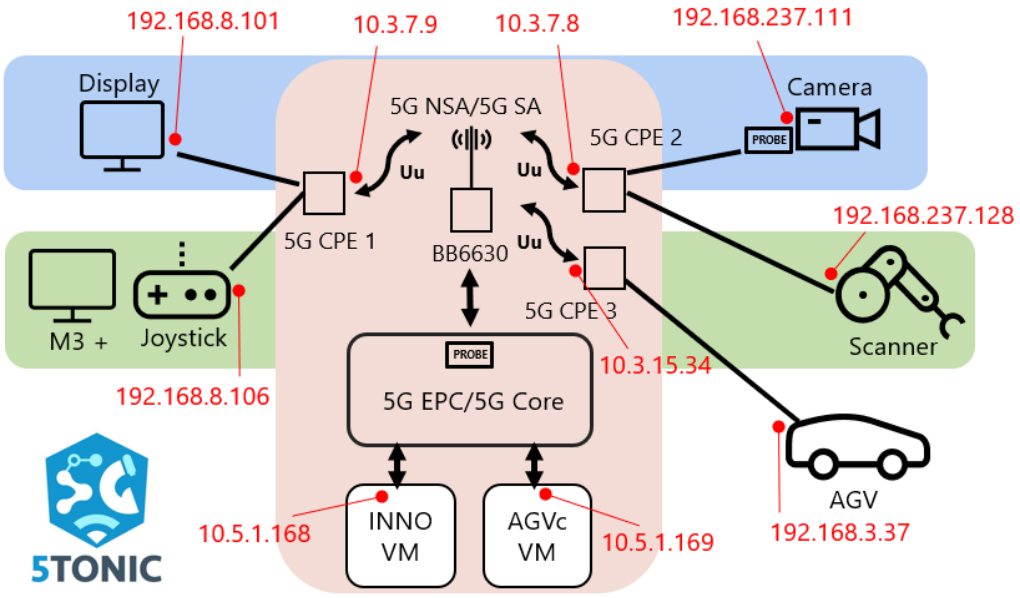

FIGURE 29: INNOVALIA UC2 SETUP

As the first step towards the full UC2 implementation, it is required to validate the operation of the AGV controlled by the AGVc VM deployed on the 5Growth platform. In this experimental session, the traffic ongoing between the AGV and the AGVc VM while the AGV is moving has been monitored, to analyse the traffic pattern and extract conclusions.

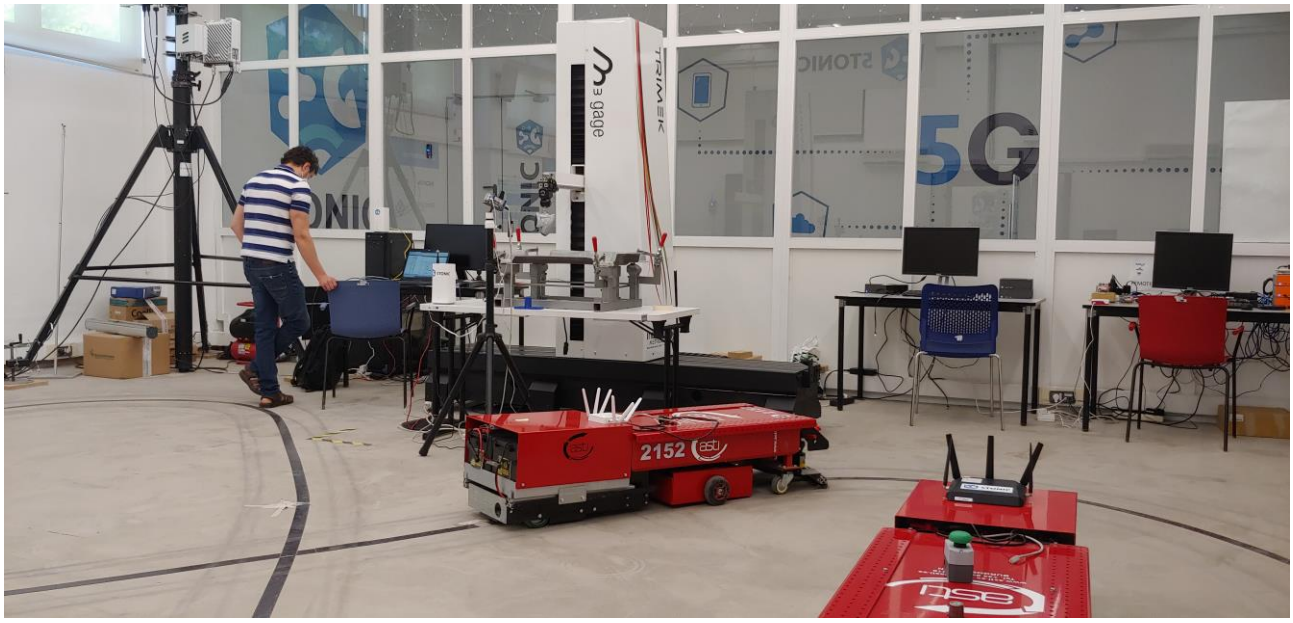

FIGURE 30: AGVS WORKING AT 5TONIC LAB

To find the value of the RTT , pings from the AGVc VM towards the AGV were performed for a period of time of over 5 minutes. The results show an average value of 16 ms and are evidenced in figure below:

```
Reply from 10.3.15.34: bytes=32 time=12ms TTL=59
Reply from 10.3.15.34: bytes=32 time=17ms TTL=59
Reply from 10.3.15.34: bytes=32 time=14ms TTL=59
Reply from 10.3.15.34: bytes=32 time=14ms TTL=59
Reply from 10.3.15.34: bytes=32 time=16ms TTL=59
Reply from 10.3.15.34: bytes=32 time=12ms TTL=59
Reply from 10.3.15.34: bytes=32 time=15ms TTL=59

Ping statistics for 10.3.15.34:
    Packets: Sent = 388, Received = 388, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 67ms, Average = 16ms
Control-C
^C
C:\Users\asti>
```

FIGURE 31: AGVC TO AGV PINGS OUTCOME

The traffic between the AGVc VM and the AGV while the AGV is moving is UDP, and it is demanding an average data rate of 450 kbps in uplink (from the AGV towards the AGVc VM) and 170 kbps in downlink (from the AGVc VM towards the AGV)

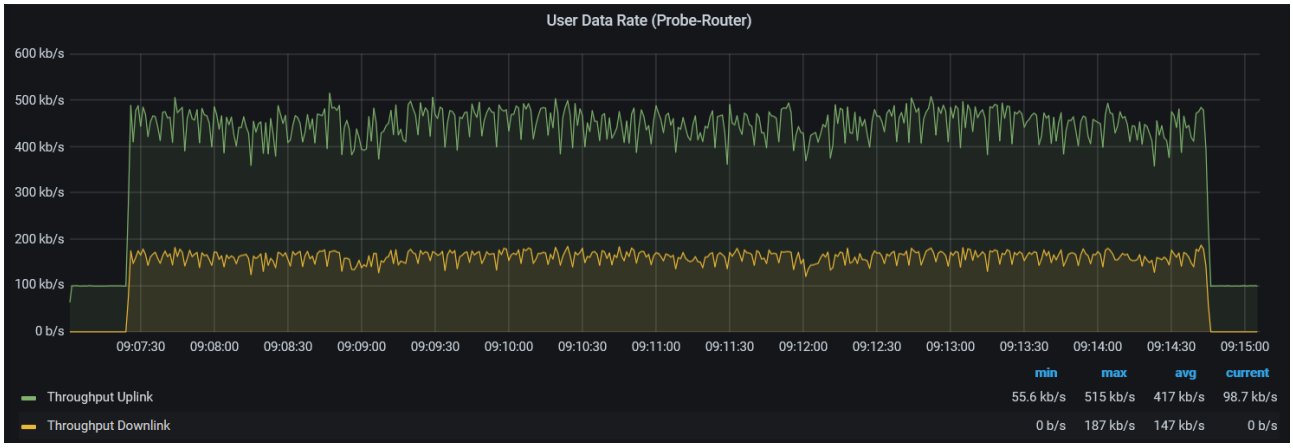

FIGURE 32: USER DATA RATE OF AGV RELATED TRAFFIC

The summary of the obtained results are the following:

TABLE 10: INNOVALIA UC2 AGV TRAFFIC MEASUREMENTS

SKPI	CKPI	CKPI result	SKPI result
P1UC2-SKPI-1: Service Operation Time (=5GR-SKPI-8)	CKPI-1 End-to-end Latency	16 ms	1 minute. (This is the time that takes the AGV to go from the starting point of the circuit to the mark that defines the destination)
	CKPI-2 Packet Loss	0 %	
	CKPI-3 Guaranteed Data Rate	450 kbps UL 170 kbps DL	
P1UC2-SKPI-4: AGV-Edge Control Synchronization (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	16 ms	Good.
	CKPI-2 Packet Loss	0 %	
P1UC2-SKPI-5: Network support for user mobility (=5GR-SKPI-3)	CKPI-2 Packet Loss	0 %	Success. (It is validated that the mobility is supported for the AGV moving in the area of coverage of the 5G cell)
	CKPI-3 Guaranteed Data Rate	450 kbps UL 170 kbps DL	
	CKPI-5 Availability	-	

Further work is ongoing to have the targeted coordinated operation of the AGV and the CMM. The observed behaviour of the integrated applications will be reported in next validation campaign.

4.2. Industry 4.0 Pilot – COMAU

This section includes the results obtained of the testing performed around the use cases within the COMAU pilot. The measurements were performed in a testbed, whose setup is shown in Figure 33, that has seen the introduction of two PC Engines APU2C4 embedded boards (APU_108 and APU_109) at either ends of the mobile infrastructure (i.e., from the 5G CPE to the radio core network server), connected to the infrastructure devices by Gigabit Ethernet links (1 GE).

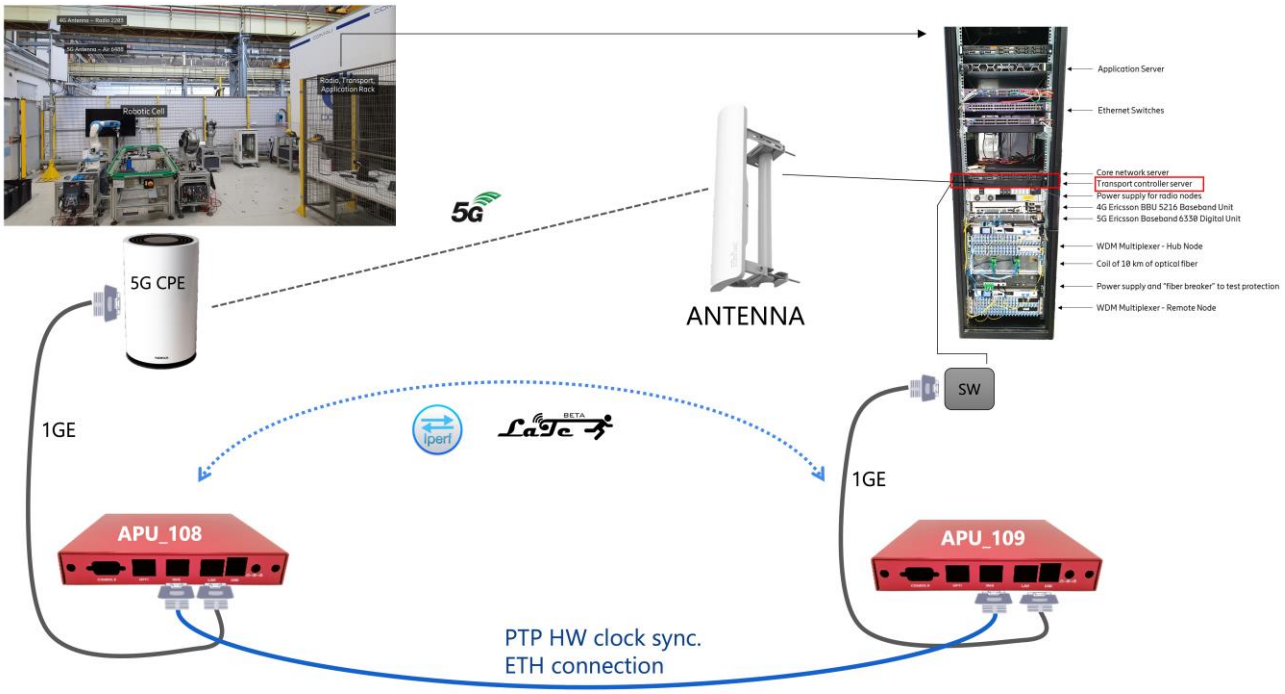

FIGURE 33: MEASUREMENT SETUP IN THE COMAU MOBILE INFRASTRUCTURE

On May 2021, the transport infrastructure, based on an optical fiber ring with DWDM technology, has been integrated in the pilot. Therefore, the performance values measured in the tests are now inclusive of the contribution of the transport layer. In the previous validation campaign, reported in D4.2, the transport layer was evaluated separately from radio by performing measurements in the Ericsson (TEI) laboratories in Pisa, Italy. Now, the radio signal is conveyed through the final transport infrastructure constituted by a fiber ring of 10 km with the related optical add/drop nodes constituting a realistic scenario in terms or radio-transport interaction.

The new test session has been conducted with the purpose of confirming the results previously obtained and to evaluate a possible change of the performance caused by the upgrade of the transport network. The test setup and parameters are the same used in the previous campaign and reported in previous deliverables.

The tests reported in this deliverable lasted one week, from June 11th to June 18th. As it was done in past measurements, the APU2C4 measurement boards continuously alternate between iPerf and LaTe measurement. iPerf 2.0.13 was used to measure the throughput in UL and DL, both using UDP and TCP, while measurements were obtained for RTT latency, DL latency, UL latency and packet loss

using LaTe. To avoid synchronization, probe packets were sent with a random exponential periodicity with minimum value 100 ms and mean 110 ms, with a new periodicity drawn every 10 packets.

4.2.1. Use Case 1: Digital Twin Apps

The main metric associated to the Use Case 1 is the latency, which is crucial to ensure a perfect alignment between the real robot and its digital replica.

Latency measurements are based on the Latency Measurement Protocol (LaMP), an application-layer protocol which can be encapsulated inside any lower-layer protocol. The protocol addresses the need for a lightweight framework which encapsulates the basic information needed to perform accurate latency measurements that are completely agnostic of the communication sublayers. The LaTe tool supports LaMP over UDP/IP to perform measurements under different conditions.

In the tests performed for UC1, the latency was evaluated using LaTe and encapsulating LaMP in a UDP payload of size 24 B. RTT, DL and UL delays were measured using the same kind of probe packets travelling in both directions, but the actual measurements of each of these quantities was carried out independently. As the pilot is on the shop floor of real plant in operations and other use cases are served by the same radio network, the validation of the latency has been done in ad hoc "measurement windows" to avoid that interfering traffic could degrade the latency performances. During an active measurement window, each latency test ran for 18 seconds, involving the exchange of, roughly, 165 packets (packet transmission times have a little random jitter added to avoid synchronization issues). The measurement results between June 12 and 17, 2021 are shown in Figure 34.

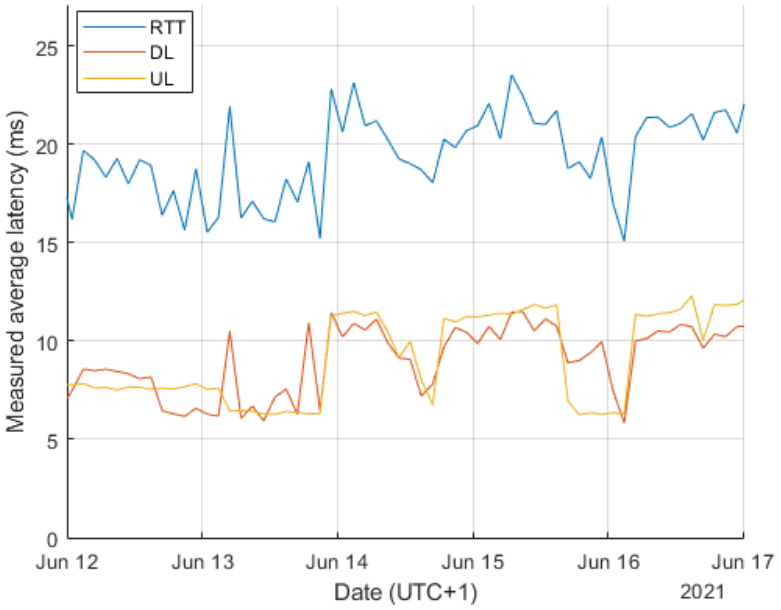

FIGURE 34: AVERAGE LATENCY

Over the whole measurement week, the average RTT was 18.23 ms, the average DL latency was 8.77ms and the average UL latency 8.5 ms. It should be noted that RTT > UL+DL, as well as the

discrepancy with previous measurements shown in D4.2 are likely due to internal operations in the CPE and in the EPS switches. In a future campaign of measurements, it will be estimated the contribution of the CPE to the measured latency by also considering the use of CPEs of different vendors. It is expected that alternative CPEs could improve the overall latency performances.

As shown in Table 11, the packet loss also plays a role in CKPI-2. The packet loss was recorded by LaTe while measuring the latency during each 18-second test (over 160-170 packets). The number of tests that failed due to a single-packet (out of 165) timeout were 3 out of 9289, leading to a packet loss of less than $2*10^{-6}$.

Finally, the TCP throughput was measured by iPerf with a socket application buffer size of 1000B and is shown in Figure 35 (downlink) and Figure 36 (uplink). Each plot shows the evolution of each metric during the measurement windows in the whole testing week. The plots report the values measured in consecutive 9-second "probe tests", at the end of which the throughput was averaged and the corresponding value inserted in the plot. It should be noted that the TCP transient was removed from each probe test (i.e., first two seconds of each probe test) in order to focus on the steady state of the network behaviour. Overall, the TCP DL average throughput measured over the week was approximately 789 Mbit/s, while the TCP UL average throughput measured over the week was around 43 Mbit/s.

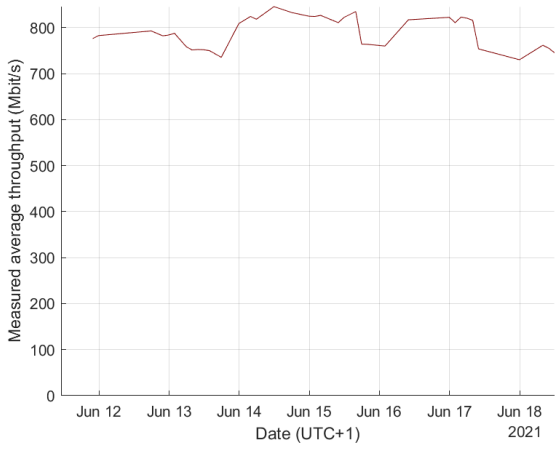

FIGURE 35: DOWNLINK TCP THROUGHPUT

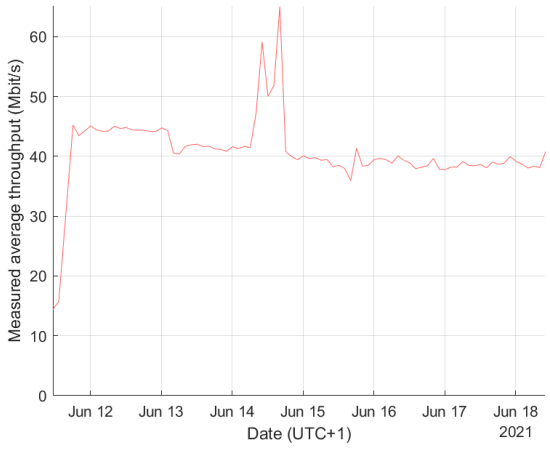

FIGURE 36: UPLINK TCP THROUGHPUT

The UDP throughput is shown in Figure 37 (downlink) and Figure 38 (uplink). The size of UPD packets was fixed at 1000B, which yielded a higher throughput than TCP, but more oscillations.

The UDP DL average throughput over the week was around 855 Mbit/s, while the UL average throughput over the week was around 42 Mbit/s.

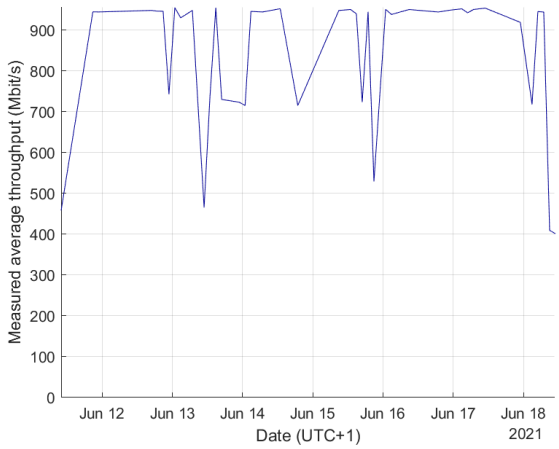

FIGURE 37: DOWLINK UDP THROUGHPUT

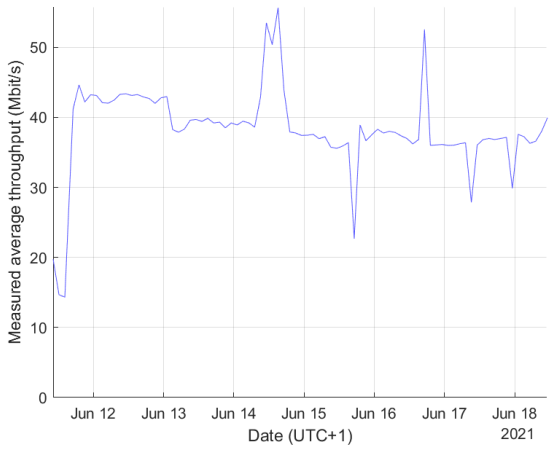

FIGURE 38: UPLINK UDP THROUGHPUT

Table 11 reports the validation results associated to UC1.

TABLE 11: COMAU UC1 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI Result	SKPI Result
P2UC1-SKPI-1 – Delay guarantees (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	18.23 ms	Good
	CKPI-2 Packet Loss	$<2*10^{-6}$	Good
P2UC1-SKPI-2 – Support of industrial protocols over 5G networks (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	18.23 ms	Good
	CKPI-2 Packet Loss	$<2*10^{-6}$	Good
P2UC1-SKPI-3 – Extensive on-plant coverage (=5GR-SKPI-7)	CKPI-1 End-to-end Latency	18.23 ms	Good
	CKPI-2 Packet Loss	$<2*10^{-6}$	Good
	CKPI-3 Guaranteed Data Rate	789.7 Mb/s (TCP-DL) 43 Mb/s (TCP-UL) 855.2 Mb/s (UDP-DL) 42 Mb/s (UDP-UL)	Good
	CKPI-5 Availability	See section 3.2.1	
	CKPI-7 Connection Density	See section 3.2.1	

4.2.2. Use Case 2: Telemetry/Monitoring Apps

Table 12 reports the validation results associated to UC2.

TABLE 12: COMAU UC2 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI Results	SKPI Results
P2UC2-SKPI-1 – Extensive coverage for high density sensors (=5GR-SKPI-7)	CKPI-1 End-to-end Latency	18.23 ms	Good
	CKPI-2 Packet Loss	$<2*10^{-6}$	Good
	CKPI-3 Guaranteed Data Rate	789.7 Mb/s (TCP-DL) 43 Mb/s (TCP-UL) 855.2 Mb/s (UDP-DL) 42 Mb/s (UDP-UL)	Good
	CKPI-5 Availability	See section 3.2.1	
	CKPI-7 Connection Density	See section 3.2.1	

4.2.3. Use Case 3: Digital Tutorial and Remote Support

Use Case 3 has jitter among its Core KPIs (CKPI-9), reported in Table 13. Thus, we have evaluated the latency variations that concur to the jitter evaluation. The effect of jitter on the video playback is usually seen in a stuttering effect either in the video or in the audio, and it can be offset using buffering at the receiver. To preserve the immediacy and interaction required by live video feeds, buffering should not exceed 100ms. Thus, it is important that the jitter remains below this value. In Figure 39 we show the average standard deviation of the latency in each batch of tests. The average standard deviation over the week was: 8.62 ms (RTT), 7.04 ms (DL), 6.33 ms (UL).

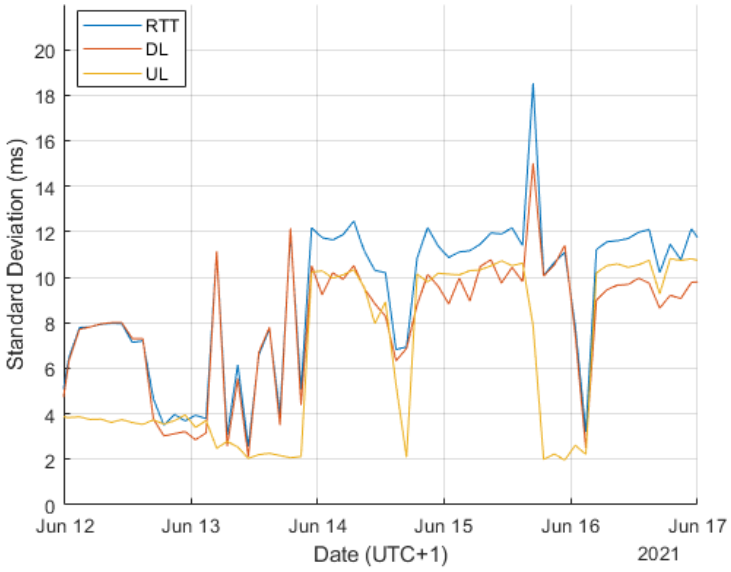

FIGURE 39: STANDARD DEVIATION OF LATENCY

Table 13 reports the validation results associated to UC3.

TABLE 13: COMAU UC3 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI Results	SKPI Results
P2UC3-SKPI-1 – High-resolution Real-time Video Quality (=5GR-SKPI-4)	CKPI-2 Packet Loss	$<2*10^{-6}$	Good
	CKPI-3 Guaranteed Data Rate	789.7 Mb/s (TCP-DL) 43 Mb/s (TCP-UL) 855.2 Mb/s (UDP-DL) 42 Mb/s (UDP-UL)	Good
	CKPI-9 Jitter	Standard deviation of latency: 8.62 ms (RTT), 7.04 ms (DL), 6.33 ms (UL)	Good
P2UC3-SKPI-2 – Network Support for Device Mobility (=5GR-SKPI-3)	CKPI-5 Availability	See section 3.2.1	
	CKPI-10 Received Radio Signal Quality	See section 3.2.1	
	CKPI-11 Buffer Occupancy	See section 3.2.1	

4.3. Transportation pilot - EFACEC_S

This section presents the results obtained in the validation campaign according with the strategy described in 3.3.1. For each Use Case, a diagram of the deployed testbed is updated and the achieved measurements are shown by tables, histograms and graphs.

Unfortunately, it was not possible to finalize the civil work and EFACEC_S validation campaign at Aveiro harbor on time to include in this deliverable. The civil work is mainly the optical fiber and energy connections between the bottom and the top of the tower on which the 5G antennas are installed, during the first week of July the civil work will be finalized. Afterwards de validation campaign will be carried out. Regarding this drawback we decided to carry out measurements at IT Lab environment and show the evolution since the last validation campaign.

4.3.1. Use Case 1: Safety Critical Communications

As described in section 3.3.1, the first considered scenario is the one related to ITAV2 (lab environment) with the complete solution (software and hardware) of the use case 1.

The next figure represents the lab and vertical premises environment for EFACEC_S UC1 supported by 5G SA network.

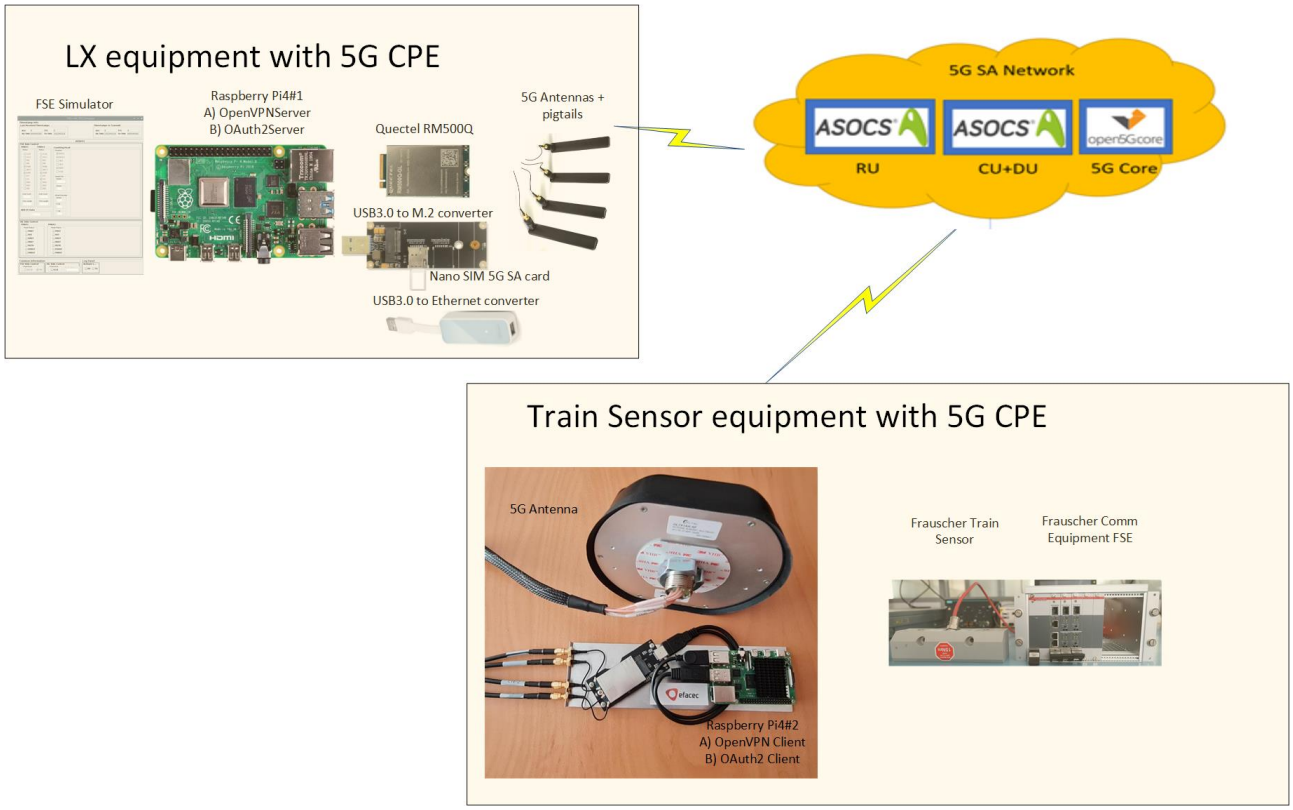

FIGURE 40: LAB AND VERTICAL PREMISES ENVIRONMENT FOR EFACEC_S UC1 SUPPORTED BY 5G SA NETWORK

This solution involves a 5G communication between the train detectors and the level crossing controller and it is composed with the following main components:

a) Train sensor detectors
b) Processor boards (Frauscher Com equipment for signaling purposes)
c) 5G-CPE (includes a Rasperry Pi device and the 5G interface); this is a custom EFACEC_S, allowing software functionalities such as monitoring, configuration, protocol implementation regarding safety and security and integration). The 5G interface is supported by the integration of Quectel modules (figure 1): This solution is used both for train detectors as well for the level crossing controller. The figure illustrates the different antenna solutions according to the environment (lab environment represented in the top of the figure and external solution in the bottom of the figure). The figure also illustrates the 5G CPE components and the 5G CPE integrated solution (bottom).
d) Level Crossing Controller – At Lab environment a simulator is used to emulate the level crossing controller (in order to validate the protocols, the safety communications requirements and to process the received information)

Next figure illustrates, in block diagrams, the logical architecture of the solution.

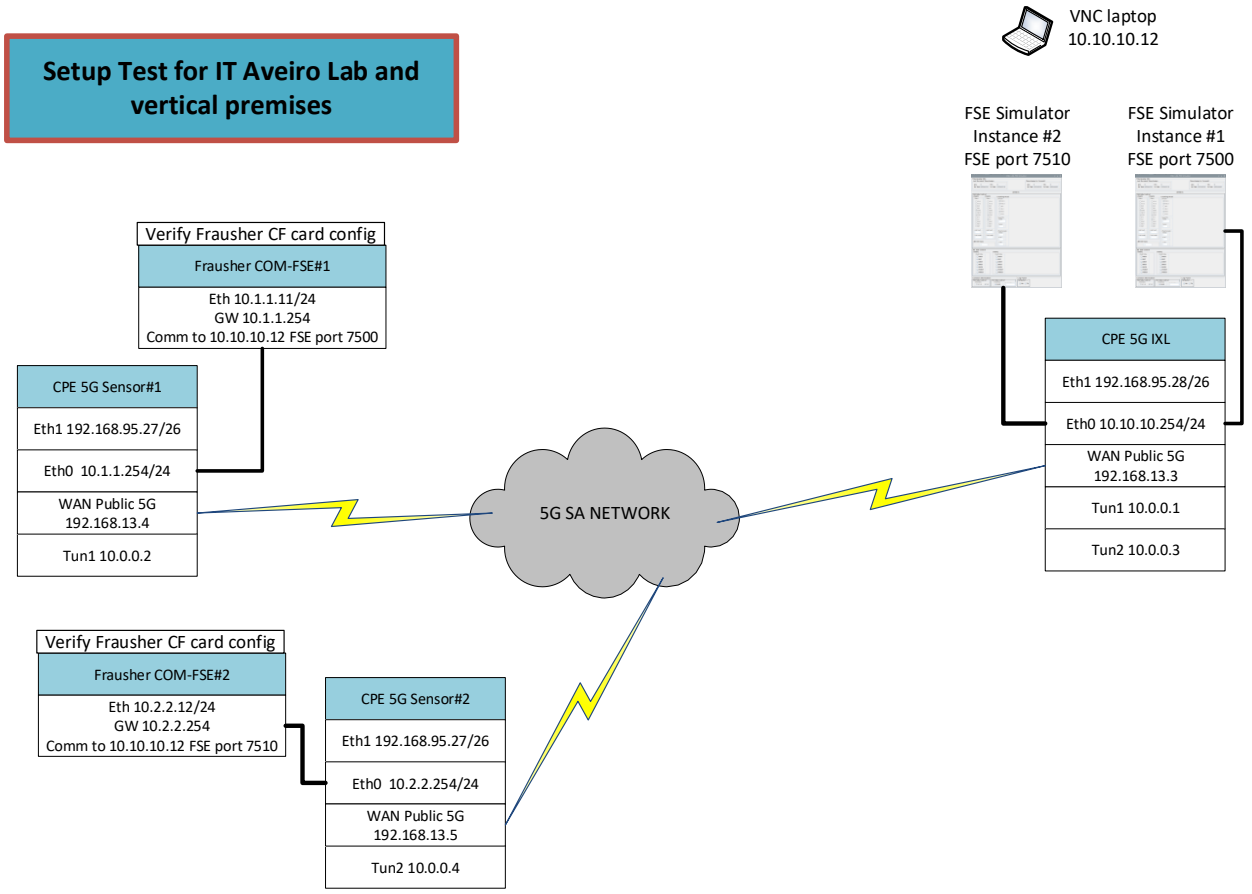

FIGURE 41: EFACEC_S UC1 SETUP USING 5G SA NETWORK

The results obtained for the UC1 are summarized in the following figures and tables.

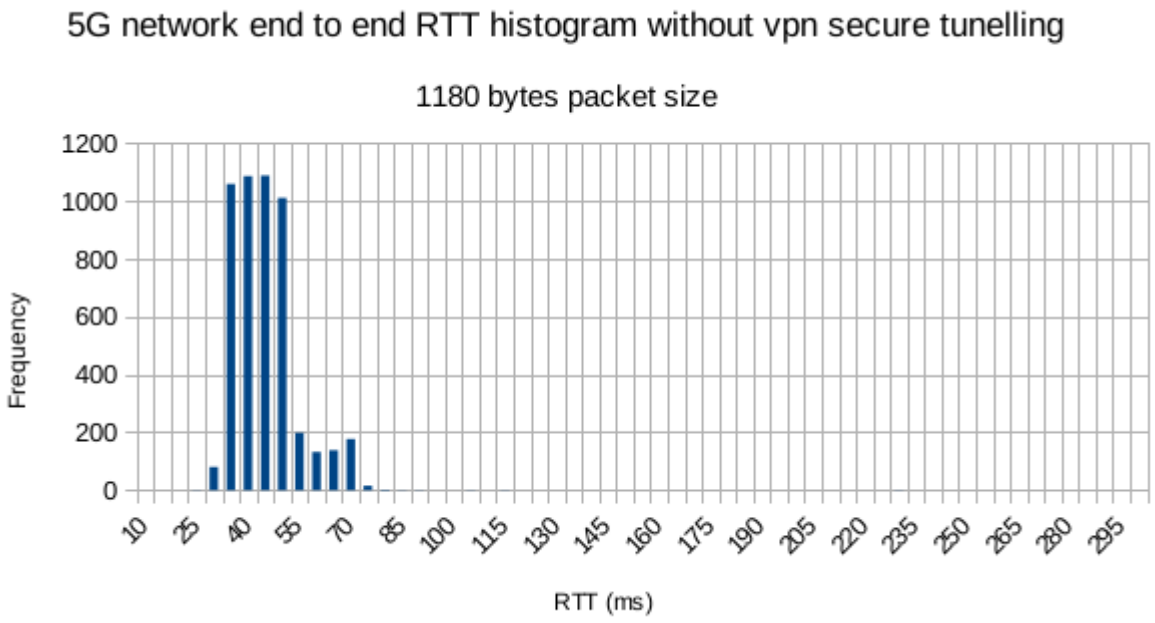

FIGURE 42: HISTOGRAM OF RTT

Regarding the performance tests, also throughput values were collected, and next tables shows the measurements on the UDP and TCP bitrates.

TABLE 14: THROUGHPUTS MEASURED FOR UDP PROTOCOL IN EFACEC_S UC1

Test	Bandwidth	Jitter (ms)	Lost/Total Datagrams	
UDP-250Kb/s	250 Kb/s	6.305	0/697	0%
UDP-500Kb/s	500 Kb/s	8.522	0/1392	0%
UDP-1Mb/s	1,00 Mb/s	9.834	0/2787	0%
UDP-2Mb/s	2,00 Mb/s	5.583	0/5572	0%
UDP-4Mb/s	4,00 Mb/s	3.658	0/11108	0%
UDP-8Mb/s	8,00 Mb/s	1.338	27/22250	0,12%
UDP-16Mb/s	16,0 Mb/s	0.823	299/44493	0,67%
UDP-32Mb/s	32,0 Mb/s	0.491	332/88937	0,37%
UDP-64Mb/s	58,1 Mb/s	0.198	2133/163855	1,3%
UDP-80Mb/s	59,8 Mb/s	0.096	1416/167706	0,84%

TABLE 15: THROUGHPUTS MEASURED FOR TCP PROTOCOL IN EFACEC_S UC1

Bitrate			Tx retries		Note
37,9	Mbps		309	packets	Test executed over 60 seconds

For the throughput analysis with the UDP protocol, the results are around 16 Mbps with no significant losses and with a jitter value of 0,823 ms.

The throughput analysis for the TCP protocol the results show a data rate of 37,9 Mbps over 1 minute.

When comparing the RTT results from the previous campaign it is possible to verify some improvements in latency, in the throughputs and also in packet loss.As explain in previous deliverables, the data messages (protocolar messages) related to this Use Case requires low bandwidths, meaning that no data packages are collected for this type of traffic.

TABLE 16: EFACEC_S UC1 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI result	SKPI result	Validation methodology
P3UC1-SKPI-1: Sync between LX detectors and Controller (=5GR-SKPI-2)	CKPI-1 End-to-end Latency	20 ms	Acceptable	Expressed as Bad/Acceptable/Good. This will be mapped with the measured Core KPIs to establish the range of values that imply a Bad, Acceptable or Good synchronization.
	CKPI-2 Packet Loss	0%		
P3UC1-SKPI-2: Communication Availability between Lx Detectors and Lx Controller (=5GR-SKPI-10)	CKPI-5 Availability	--	--	Ability of a product / equipment / system to be in state to perform a required function under given conditions and environment at a given instant of time or over a given interval assuming that the required external resource is provided; In fact, the goal is to

				measure the time the system will be unavailable. Availability = 1- Unavailability

The following graph shows the latency variation along the time for the UC1 solution using the 5G monitoring platform, for scales of 5 and 1 minute. According to the collected measures it be can be verified that the variation occurs around an average value of around 20 ms.

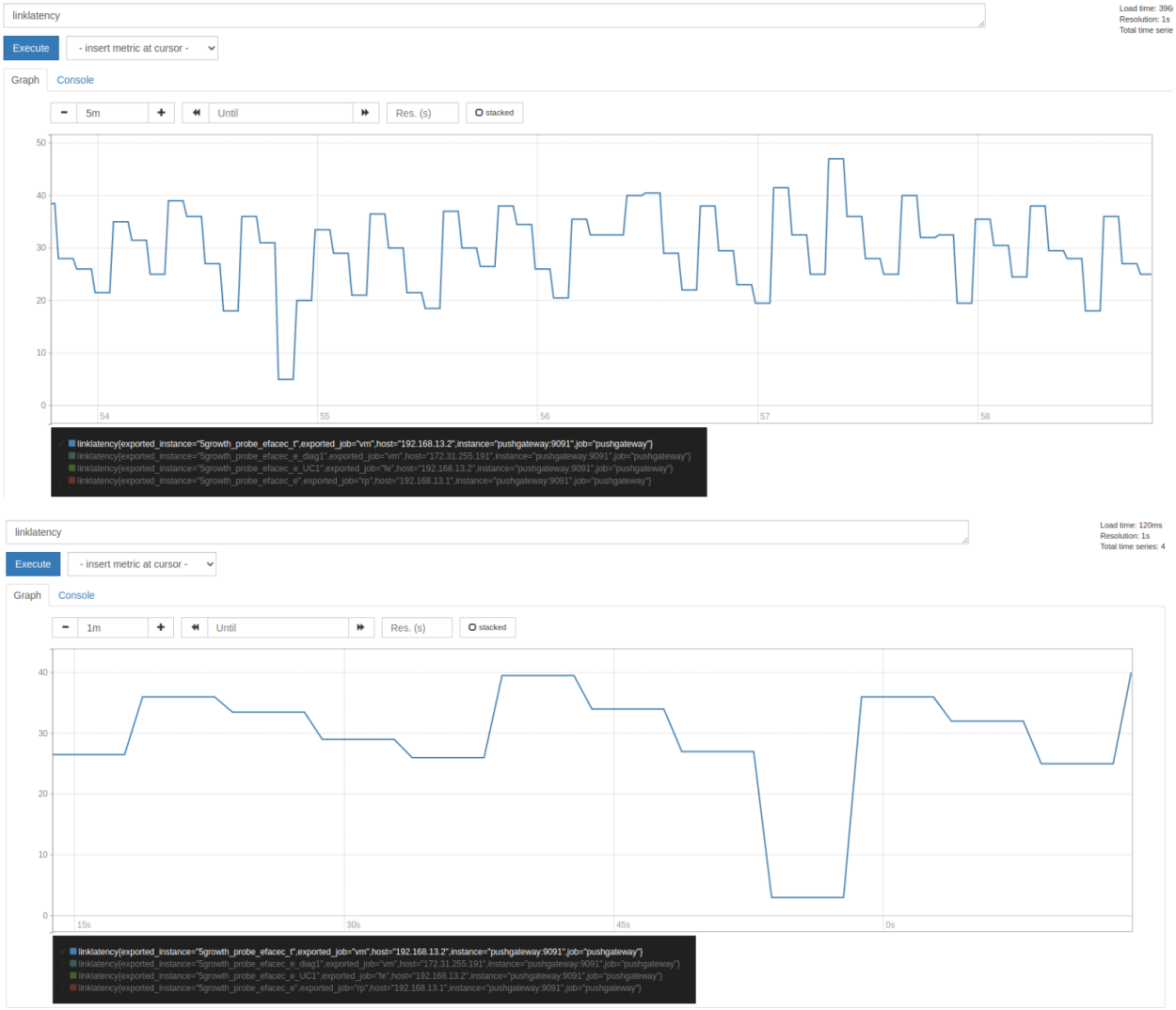

FIGURE 43: LATENCY GRAPH

Similar graphs are being collected and registered in the 5G monitoring platform, regarding jitter, packet loss (first probe integration) and throughput (second probe integration).

4.3.2. Use Case 2: Non-safety Critical Communications

Next figure represents the environment (similar for Lab and for vertical premises) related to EFACEC_S UC2, supported by 5G SA network.

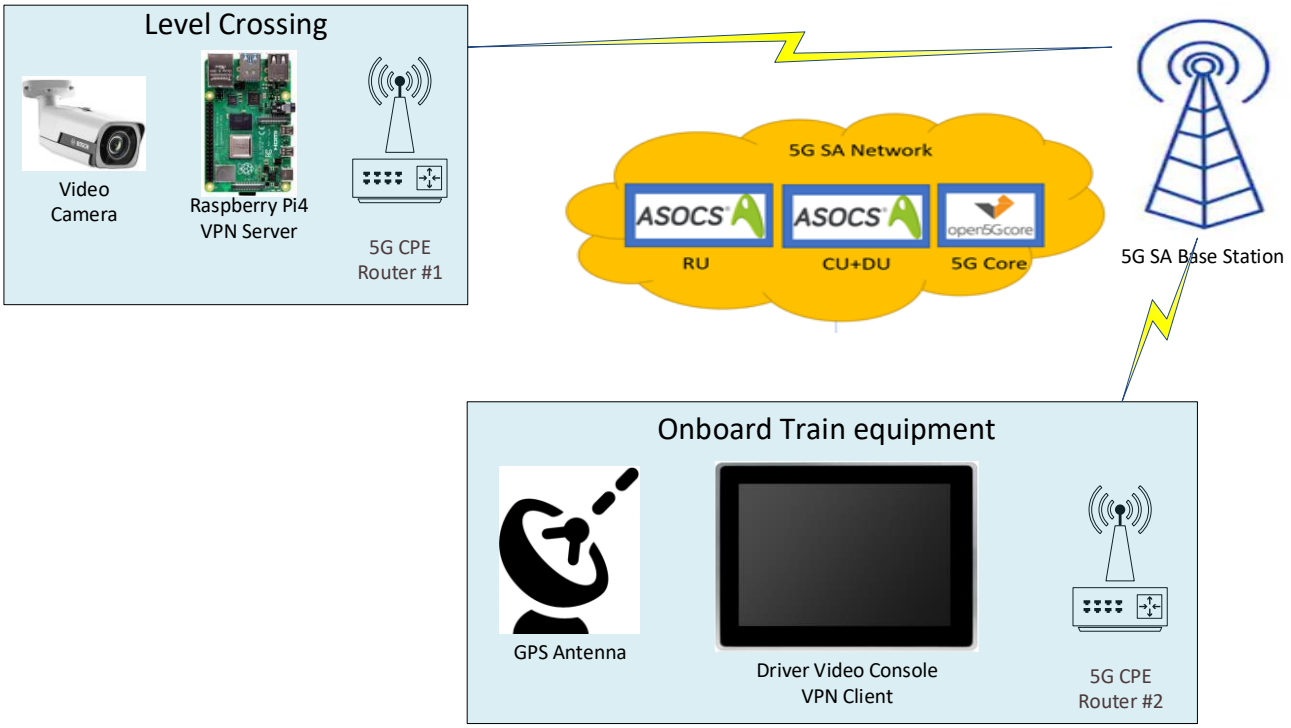

FIGURE 44: LAB AND VERTICAL PREMISES ENVIRONMENT FOR EFACEC_S UC2 SUPPORTED BY 5G SA NETWORK

This solution involves a 5G communication between a video camera covering the level crossing area and the train driver console (onboard) and it is composed with the following main components:

a) Level Crossing – HD video IP camera (Bosh), Raspberry PI device and 5G CPE (Huawei)
b) 5G Network – Radio Unit, CU + DU (ASOCS and a 5G Core (Open5GCore)
c) Train – GPS antenna (for localization purposes), Driver Console and 5G CPE (Huawei)

Next figure illustrates in block diagrams the logical architecture of the solution.

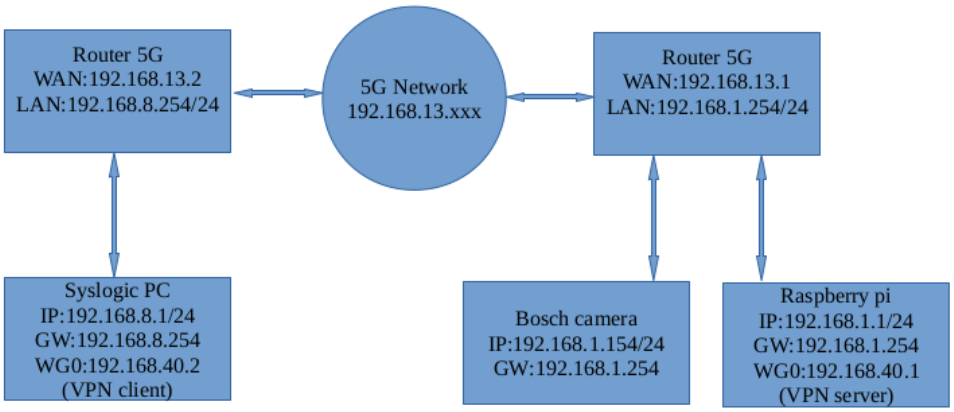

FIGURE 45: EFACEC_S UC2 SETUP USING 5G SA NETWORK

On the right side of the figure, it is represented the equipment located in the level crossing. This site includes, the 5G router, the Raspberry Pi device and a video camera. The router allows access to the

5G network, and this assigns the IP of the WAN interface to it. In the local LAN, it is integrated the video camera that will allow the acquisition of the image of the level crossing and a Raspberry pi that aims to expand the router's capacity. This Raspberry pi will allow running the part of the VPN wireguard server (security mechanism), a feature not available on the router, which will allow establishing a secure tunnel to the equipment on-board the vehicle and also does NAT for the local network. On the left side of the figure, it is represented the onboard equipment located in the vehicle/train. In the train it is integrated the 5G router and a train driver console (Syslogic PC console). The 5G router will allow access to the 5G network, and this will assign the IP of the WAN interface to it. The Syslogic PC console running Debian Linux, is where the EFACEC_S UC2 application is running to show the video and where the client wireguard secure tunnel is created, for the remote level crossing. This syslogic console is connected in the local network to the 5G router embedded in the vehicle.

The results obtained for the UC2 are summarized in the following figures and tables

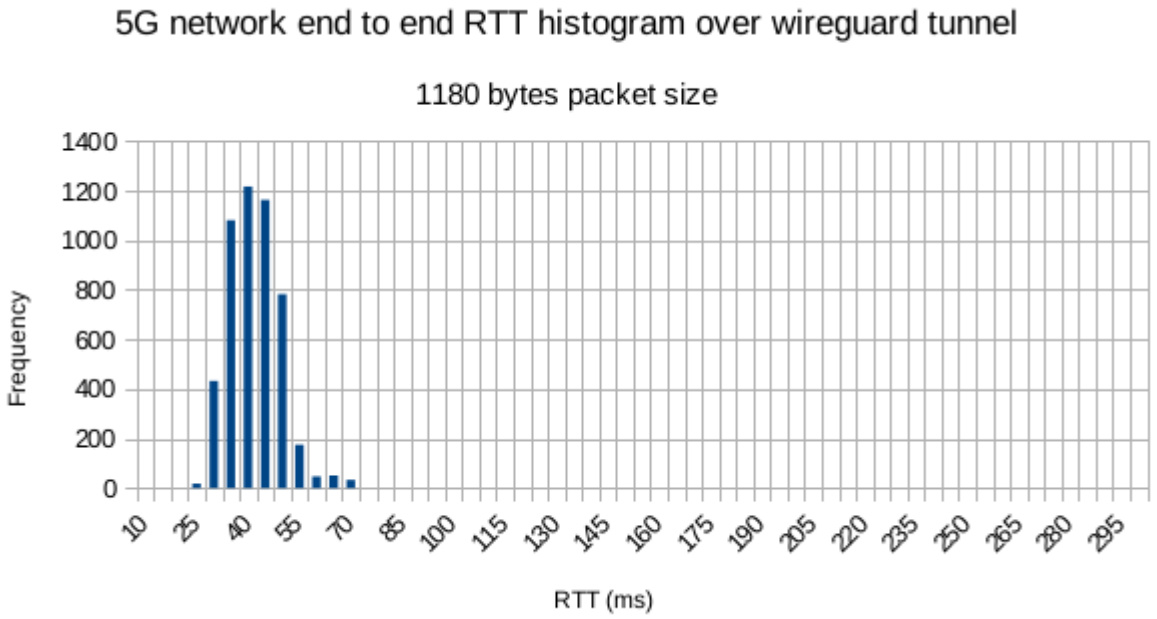

FIGURE 46: HISTOGRAM OF RTT FOR 1180 BYTES PACKET SIZE

The RTT and jitter values were measured end-to-end between the Syslogic PC console and the Raspberry pi at the other end, passing through the secure wireguard tunnel. Taking into account that in the total route, the measured RTT passes through two Up-links and two Down-links, given its symmetry, we can state without doubt that in this case the end-to-end latency is exactly half of the measured RTT.

The RTT average in this scenario is 39,556 ms and minimum values of 23,631ms was collected. The average latency value for this test scenario = 19,778 ms and minimum values of 11,816 ms was collected. This result is roughly in line with the latency tests performed between the CPE and the 5G Core (Internet Gateway) in which the average value obtained for RTT delay was roughly 50% compared to the CPE-to-CPE tests.

Next table shows the bandwidth test with iperf3 for UDP data traffic.

TABLE 17: BANDWIDTH FOR UDP TRAFFIC PROTOCOL IN EFACEC_S UC2

Test	Bandwidth	Jitter (ms)	Lost/Total Datagrams	
UDP-250Kb/s	250 Kb/s	5.509	0/1634	0%
UDP-500Kb/s	500 Kb/s	5.694	0/3268	0%
UDP-1Mb/s	1,00 Mb/s	8.902	0/6536	0%
UDP-2Mb/s	2,00 Mb/s	4.488	0/13073	0%
UDP-4Mb/s	4,00 Mb/s	4.083	0/26140	0%
UDP-8Mb/s	8,00 Mb/s	1.803	0/52283	0%
UDP-16Mb/s	16,0 Mb/s	0.648	2/104584	0,0019%
UDP-32Mb/s	32,0 Mb/s	0.303	153/209230	0,073%
UDP-64Mb/s	58,8 Mb/s	0.160	33083/417379	7,9%
UDP-80Mb/s	58,6 Mb/s	0.167	138798/521718	27%

The performance of the 5G network, in terms of UDP traffic is usable for values around 32 Mbps with 0,073% of lost datagrams. The maximum UDP throughput is around 58 Mbps but the lost datagram is high (around 7,9 %). It should be noted that the limiting factor of data throughput is the radio uplink. Upstream and downstream throughputs were measured separately, and the results were in the order of 65 Mb/s and 500 Mb/s, respectively. In any case, a more symmetrical result can be obtained through RAN configuration and will be tested in the future

TABLE 18: EFACEC_S UC2 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI result	SKPI result	Validation methodology
P3UC2-SKPI-1: High-resolution Real-time Video Quality (=5GR-SKPI-4)	CKPI-2 Packet Loss	0,073%	Fair	Expressed in values from 1 to 5. *Mean Opinion Score (MOS) is a well-known measure of video quality (5-Excellent, 4-Good, 3-Fair, 2-Poor, 1-Bad).*
	CKPI-3 Guaranteed Data Rate	32 Mps		
	CKPI-9 Jitter	0,303 ms		
P3UC1-SKPI-2: Real Time sensors monitoring latency (=5GR-SKPI-2)	CKPI-1 e2e latency	19,778 ms	Good	Expressed as Bad/Acceptable/Good. This will be mapped with the measured Core KPIs to establish the range of values that imply a Bad, Acceptable or Good synchronization.
	CKPI-2 packet Loss	0%		
P3UC3-SKPI-3: Real Time Sensors Monitoring Communication Availability	CKPI-5 Availability	--	--	Ability of a product / equipment / system to be in state to perform a required function under given conditions and environment at a given instant of time or over a given interval assuming that the required external resource is provided; In

(=5GR-SKPI-10)				fact the goal is to measure the time the system will be unavailable. Availability = 1-Unavailability.

Since the UC2 includes a GUI running on the driver console, some developments were performed in to display providing the performance values of the application. Next figure shows an image of the GUI representing in real time the latency (19,591 ms), jitter (3.129 ms) and data rate (822,375 kbps) when the HD video stream is transmitting over the 5G network.

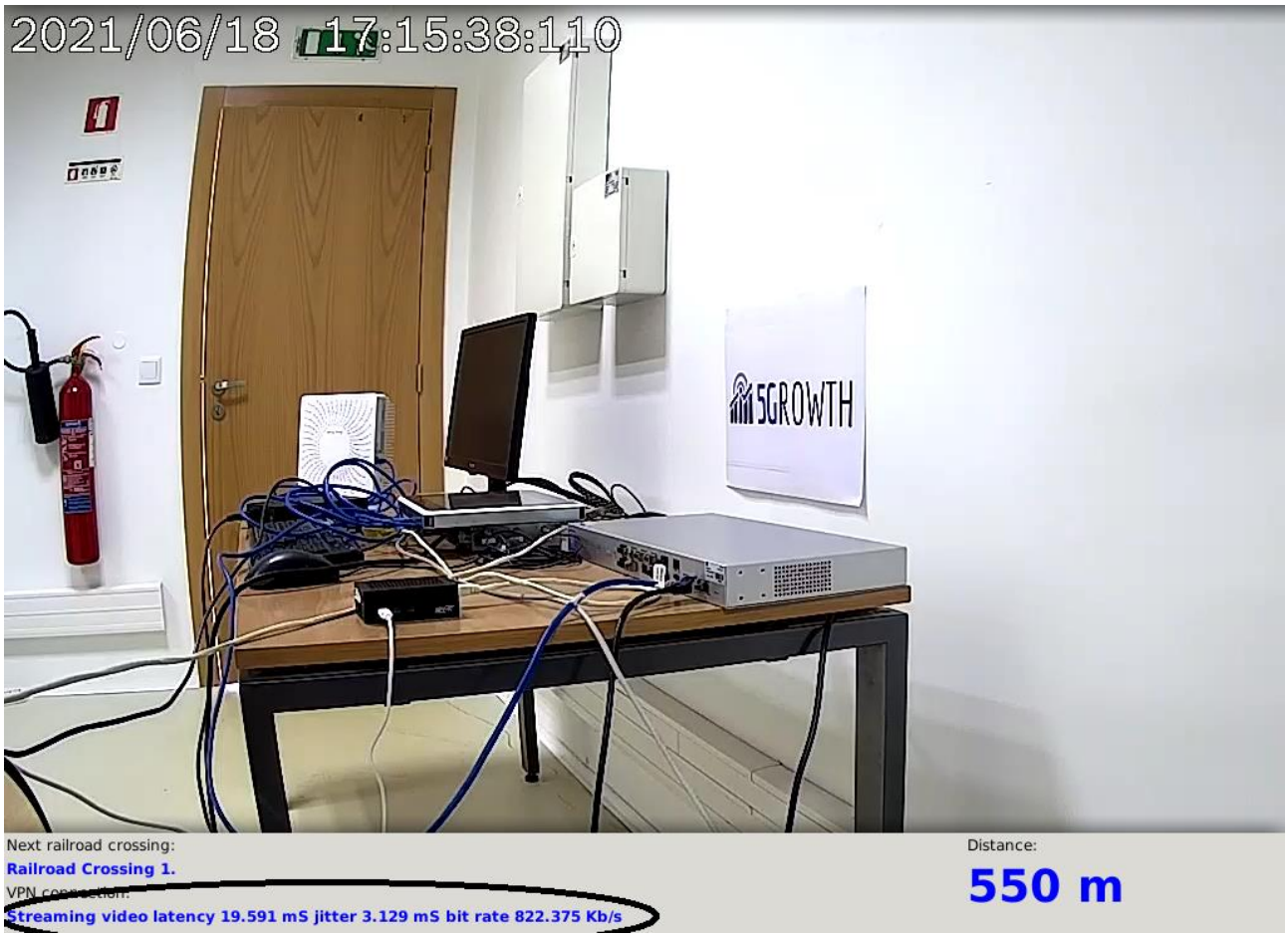

FIGURE 47: GUI OF THE TRAIN DRIVER CONSOLE

When comparing the measurements results with the previous campaign it is possible to verify some improvements in latency, in the throughputs and also in packet loss allowing this way to transmit the HD video imagens with a good quality.

4.4. Energy Pilot - EFACEC_E

This section includes the updated description of the Energy Pilot testbed deployed in IT Lab, and the most recent results obtained from tests performed around the use cases within the EFACEC_E pilot.

The recent test campaigns were carried out in the temporary testbed deployed in the IT lab as depicted below:

- Two raspberry-PI boards were added to the testbed to allow the use of SSSA probes in the measurements.
- One probe developed by ITAV was installed in the datacenter to measure the throughput in the network interfaces of the Frontend and Workstation VMs.

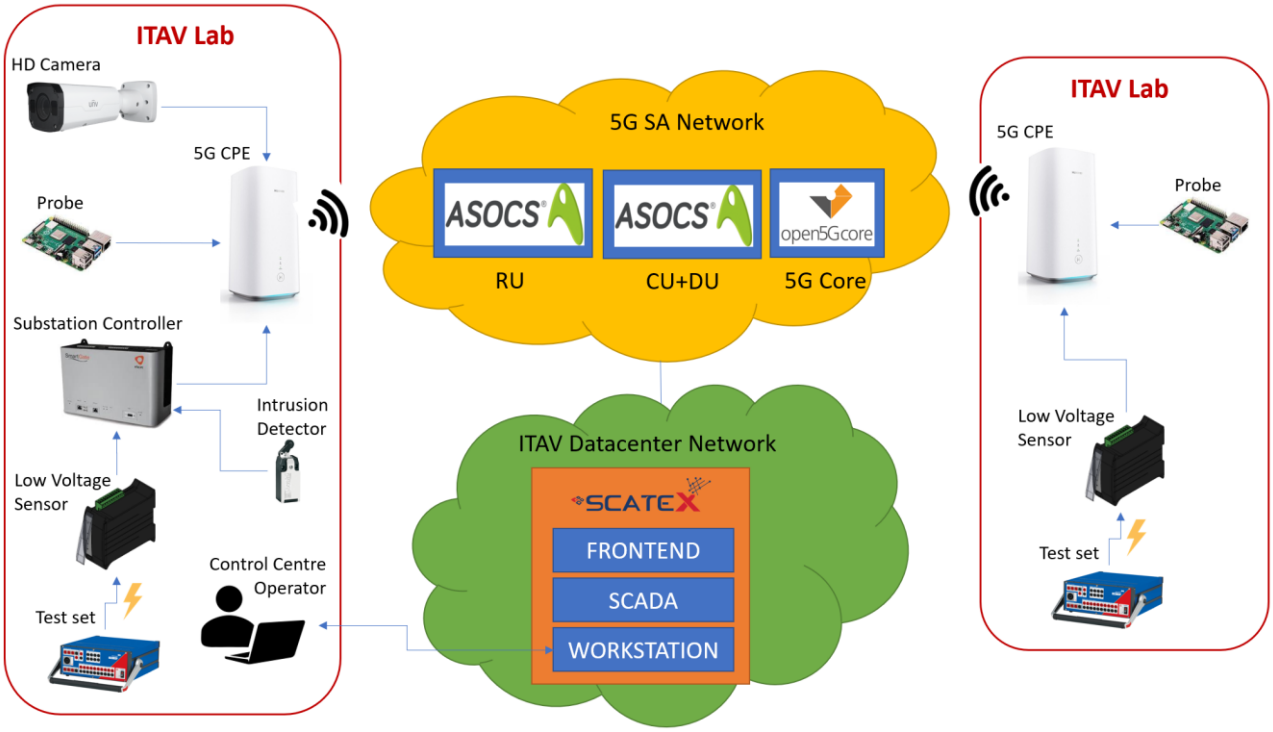

FIGURE 48: LAB ENVIRONMENT FOR EFACEC_E UC1 AND UC2 SUPPORTED BY 5G SA NETWORK

4.4.1. Use Case 1: Advanced Monitoring and Maintenance Support for Secondary Substation MV/LV Distribution Substation

The testing setup for the UC1, currently fully implemented in IT lab, is depicted in Figure 48 and described below:

- The Substation Controller (GSmart), the HD Video IP camera and the raspberry-Pi device were connected to one Huawei 5g CPE through a network switch.
- The ScateX# control center software, including the Frontend Server and the Operator Workstation were virtualized and hosted in IT Datacenter.
- The SSSA probes were installed in the raspberry-Pi device on the secondary substation side, and in the Frontend VM on the control centre side.

This setup is an evolution of the one described in D4.2, section 5.4.1.2, to incorporate the SSSA probes.

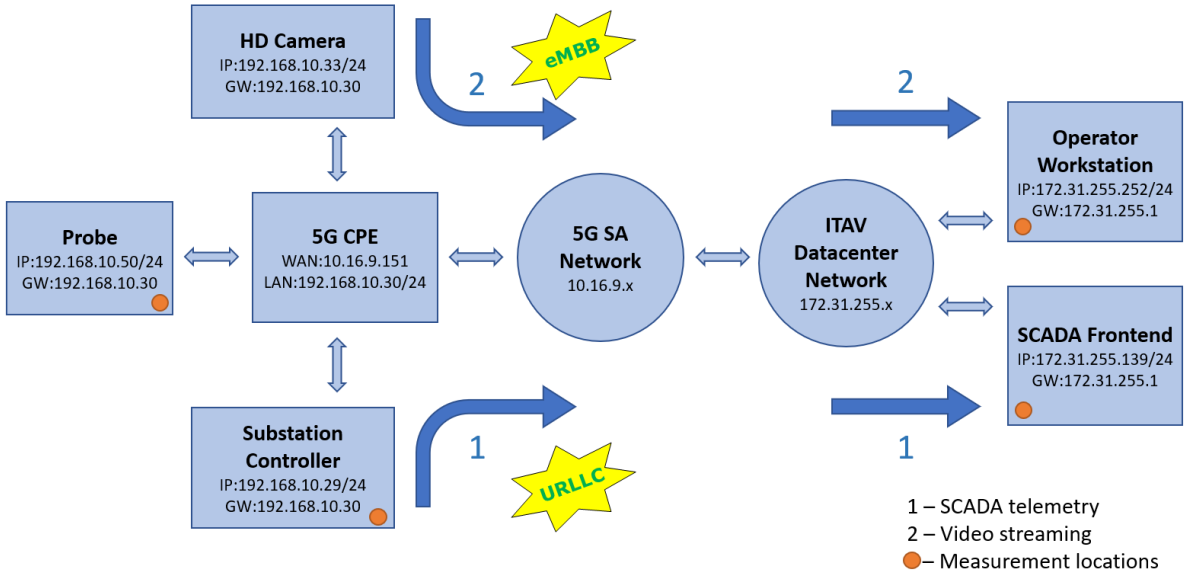

FIGURE 49: EFACEC_E UC1 SETUP USING 5G SA NETWORK WITH ASOCS RAN AND OPEN5GCORE

The results obtained for the UC1 in this campaign are summarized in the following figures and tables.

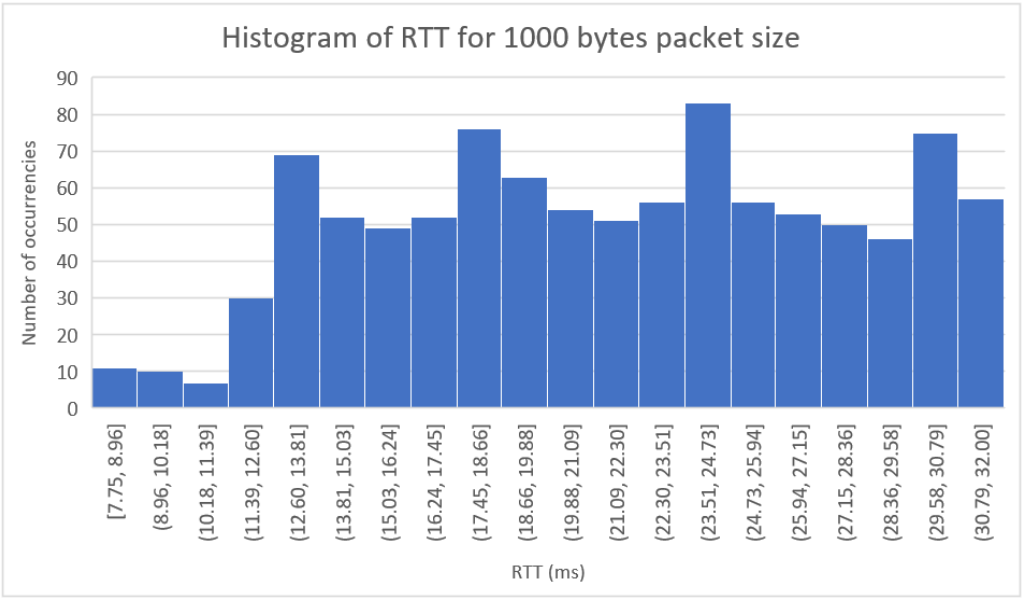

FIGURE 50: HISTOGRAM OF RTT FOR UC1 (1000 BYTES PACKET SIZE)

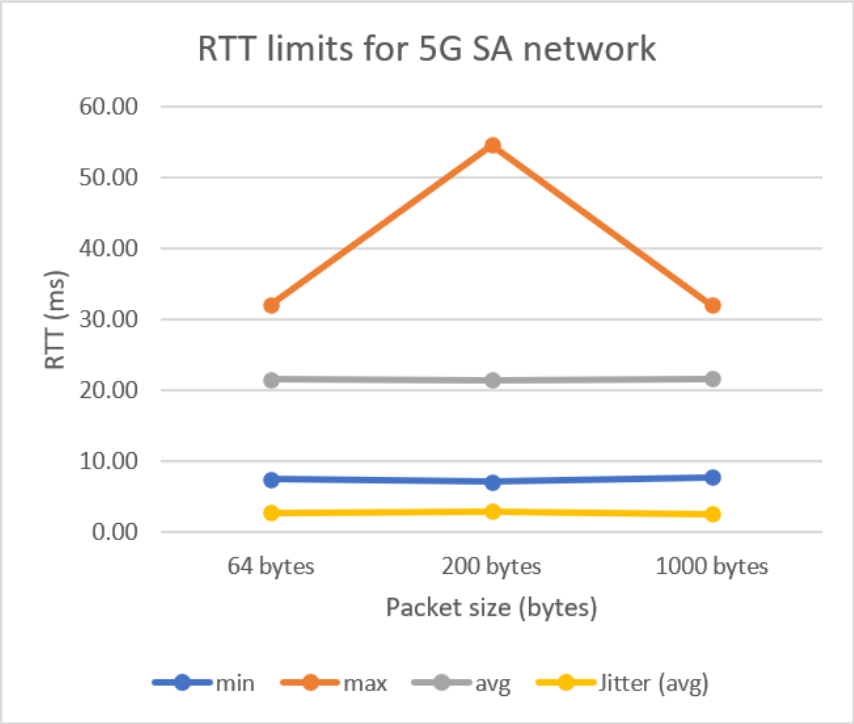

FIGURE 51: UC1 RTT LIMITS FOR DIFFERENT PACKET SIZES

TABLE 19: UC1 THROUGHPUT MEASURED FOR UDP PROTOCOL

Bitrate		Bitrate receiver		Jitter		Lost Datagrams	Total Datagrams	Lost/Total
1	Mbits/sec	1.0	Mbits/sec	1.698	ms	0	927	0.00%
10	Mbits/sec	10.0	Mbits/sec	0.243	ms	0	9273	0.00%
20	Mbits/sec	20.1	Mbits/sec	0.174	ms	0	18578	0.00%
30	Mbits/sec	30.0	Mbits/sec	0.165	ms	0	27813	0.00%
40	Mbits/sec	40.1	Mbits/sec	0.132	ms	1893	37098	5.10%
50	Mbits/sec	50.3	Mbits/sec	0.137	ms	0	46531	0.00%
60	Mbits/sec	60.1	Mbits/sec	0.131	ms	0	55653	0.00%

TABLE 20: UC1 THROUGHPUT MEASURED FOR TCP PROTOCOL

Interval	Transfer	Throughput	Tx retries	
0.00-60.00 sec	363 MBytes	50.8 Mbits/sec	1632	sender
0.00-60.00 sec	363 MBytes	50.8 Mbits/sec		receiver

TABLE 21: EFACEC_E UC1 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI Result	SKPI Result
P4UC1-SKPI-1: Monitoring Sensors Information Collection and Visualization End-to-end Latency (5GR-SKPI-2)	CKPI-1: E2E latency	11ms	Good
	CKPI-2: Packet Loss (@ ~10Mbit load)	0%	
P4UC1-SKPI-2: Monitoring Sensors Information Collection and Visualization Availability (5GR-SKPI-10 & 5GR-SKPI-2)	CKPI-2: Packet Loss (@ ~10Mbit load)	0%	Good
	CKPI-5: Availability(*)	100%	
P4UC1-SKPI-3: High-definition Surveillance Video (5GR-SKPI-4)	CKPI-2: Packet Loss (@ ~10Mbit load)	0%	Good
	CKPI-3: Guaranteed Data Rate	50Mbit/s	
	CKPI-9: Jitter (@ ~10Mbit load)	0.243ms	
P4UC1-SKPI-4: Augmented Reality Information Real-time End-to-end Latency (5GR-SKPI-2)	CKPI-1: E2E latency	--	
	CKPI-2: Packet Loss	--	
P4UC1-SKPI-5: Augmented Reality information Real-time Availability (5GR-SKPI-10 & 5GR-SKPI-2)	CKPI-2: Packet Loss	--	
	CKPI-5: Availability	--	

(*) The service availability test ran for a very limited period (5 min). Further service availability testing for longer periods shall be executed in the next measurement campaign.

As mentioned before, during this campaign it was possible to integrate two 5Growth probes for latency related KPIs and throughput. These probes are collecting data and automatically feeding it to the 5G monitoring platform.

The following graphs shows the latency and the jitter variation along the time for the UC1 solution using the 5G monitoring platform, for a period of 15 minutes. According to the collected measures it be can be verified that the variation occurs around an average value of around 20 ms.

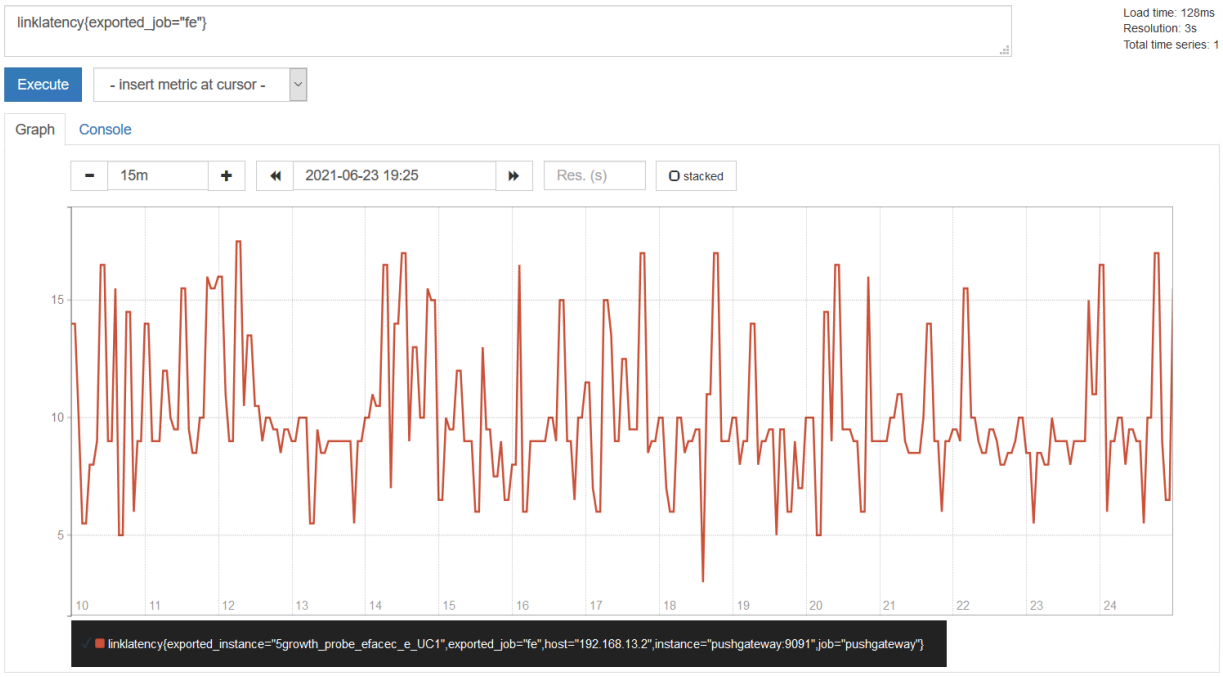

FIGURE 52: 5G MONITORING PLATFORM - UC1 LATENCY IN A 15 MINUTES PERIOD

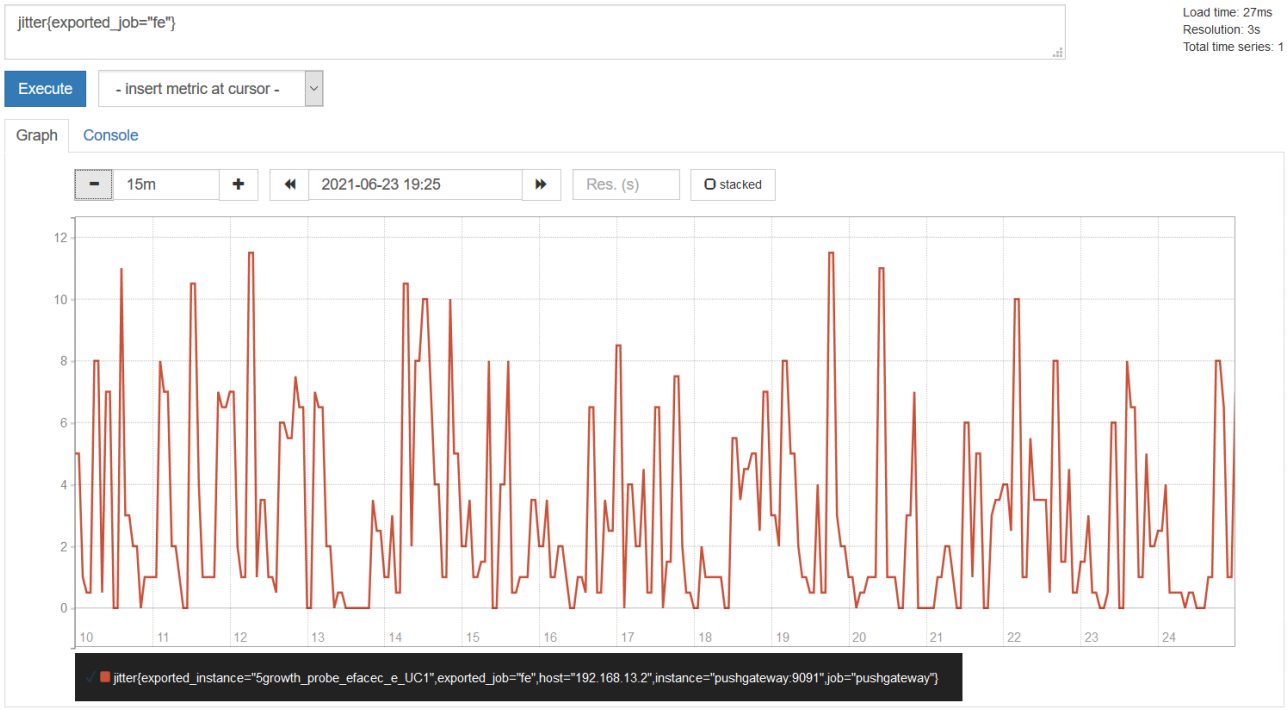

FIGURE 53: 5G MONITORING PLATFORM - UC1 JITTER IN A 15 MINUTES PERIOD

4.4.2. Use Case 2: Advanced Critical Signal and Data Exchange across Wide Smart Metering and Measurement Infrastructures

The testing setup for the UC2, currently fully implemented in IT lab, is depicted in Figure 54 and described below:

- The GSmart equipment and the first raspberry-Pi device were connected to one Huawei 5g CPE through a network switch.
- The LVS3 equipment and the second raspberry-Pi device were directly connected to one second Huawei 5G CPE.
- The SSSA probes were installed in the two raspberry-Pi devices, one on the secondary substation side, and the other on the low voltage electrical network side.

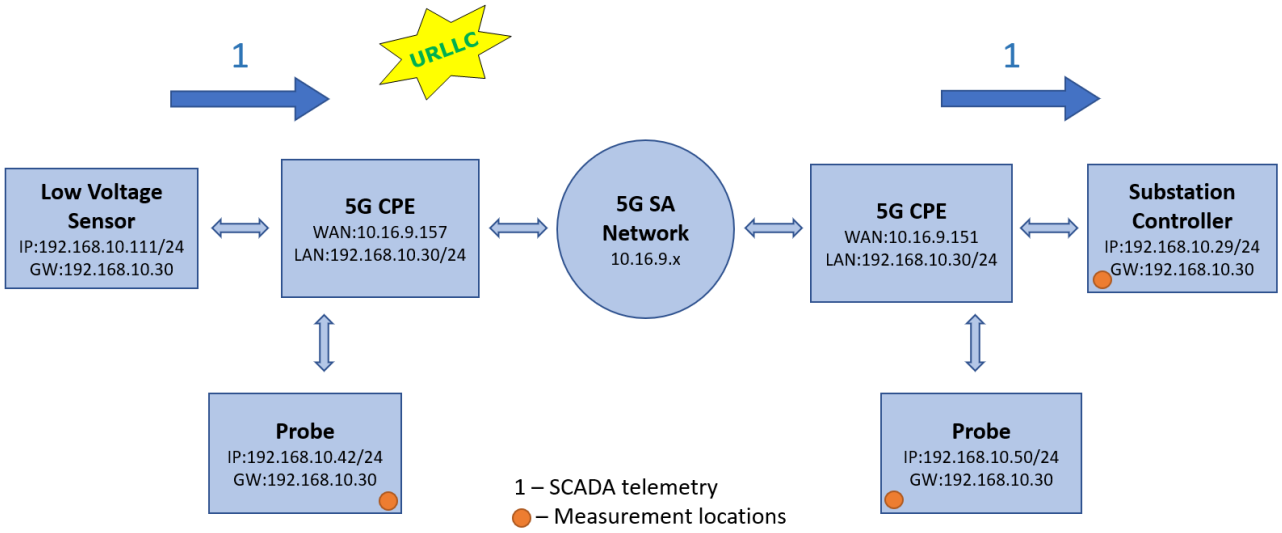

FIGURE 54: EFACEC_E UC2 SETUP USING 5G SA NETWORK WITH ASOCS RAN AND OPEN5GCORE

The results obtained for the UC2 in this campaign are summarized in the following figures and tables.

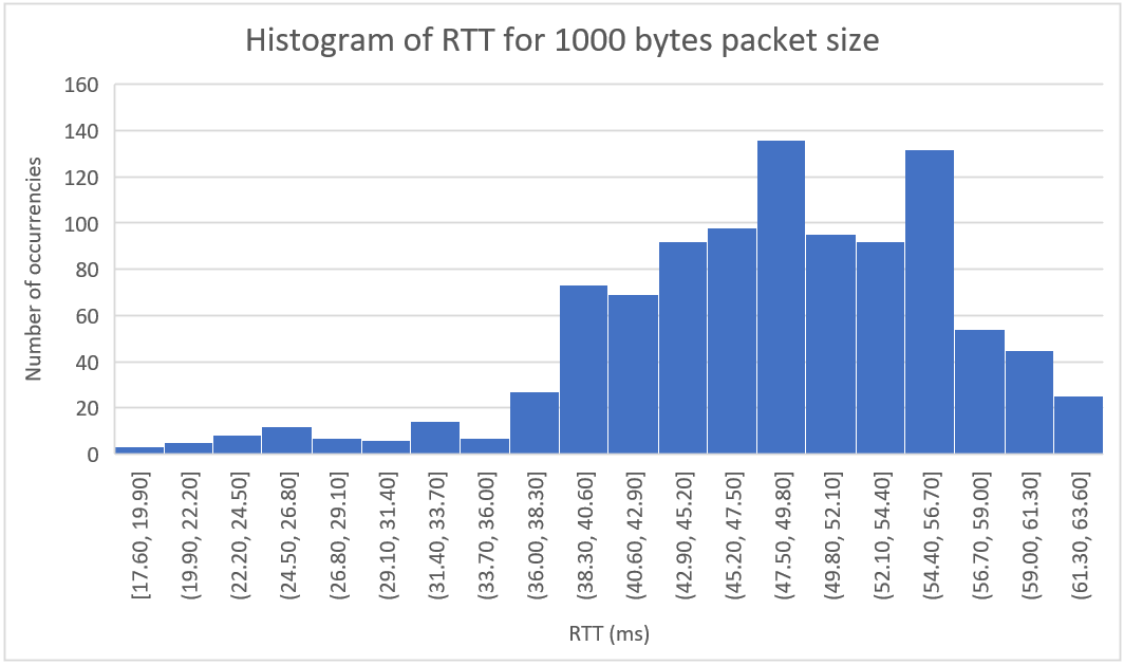

FIGURE 55: HISTOGRAM OF RTT FOR UC2 (1000 BYTES PACKET SIZE)

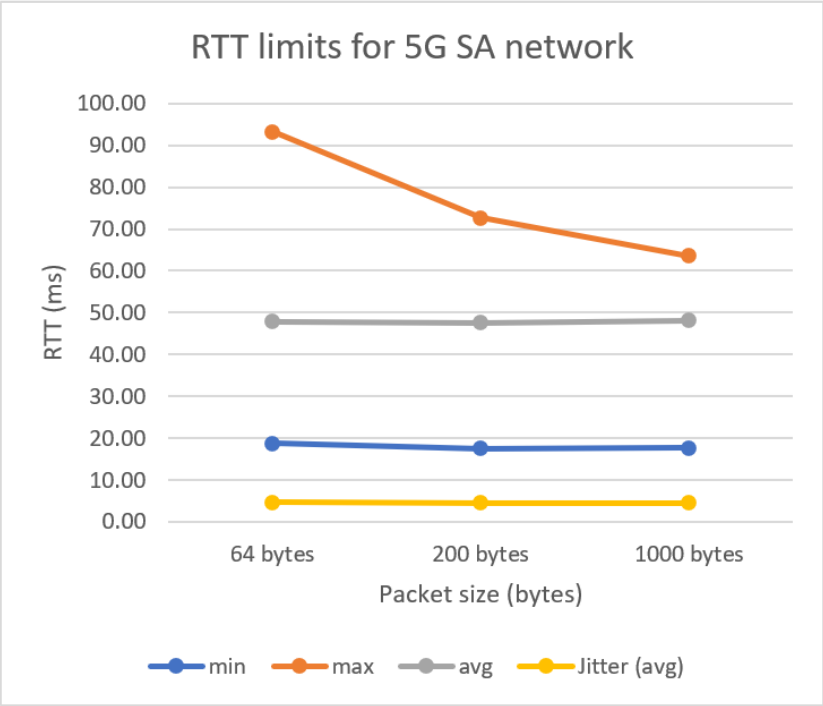

FIGURE 56: UC2 RTT LIMITS FOR DIFFERENT PACKET SIZES

TABLE 22: EFACEC UC2 SPECIFIC SERVICE KPIS, CORE KPIS AND VALIDATION METHODOLOGY

SKPI	CKPI	CKPI Result	SKPI Result
P4UC2-SKPI1: Last-gasp Information End-to-end latency (=5GR-SKPI-2)	CKPI-1: E2E latency	24ms	Poor
	CKPI-2: Packet Loss (@ ~1Mbit load)	0%	
P4UC2-SKPI2: Last-gasp Information Connectivity Availability (=5GR-SKPI-10 & 5GR-SKPI-2)	CKPI-2: Packet Loss (@ ~1Mbit load)	0%	Good
	CKPI-5: Availability	100%	
P4UC2-SKPI3: Secondary Substation End-to-end Latency (=5GR-SKPI-2)	CKPI-1: E2E latency	24ms	Good
	CKPI-2: Packet Loss (@ ~1Mbit load)	0%	
P4UC2-SKPI4: Secondary Substation Availability	CKPI-2: Packet Loss (@ ~1Mbit load)	0%	Good

(=5GR-SKPI-10 & 5GR-SKPI-2)	CKPI-5: Availability	100%	
P4UC2-SKPI5: Time Synchronization between Low Voltage Sensors (=5GR-SKPI-2)	CKPI-1: E2E latency	24ms	Poor
	CKPI-2: Packet Loss (@ ~1Mbit load)	0%	

(*) The service availability test ran for a very limited period (5 min). Further service availability testing for longer periods shall be executed in the next measurement campaign.

The following graphs shows the latency and the jitter variation along the time for the UC2 solution using the 5G monitoring platform, for a period of 15 minutes. According to the collected measures it be can be verified that the variation occurs around an average value of around 20 ms.

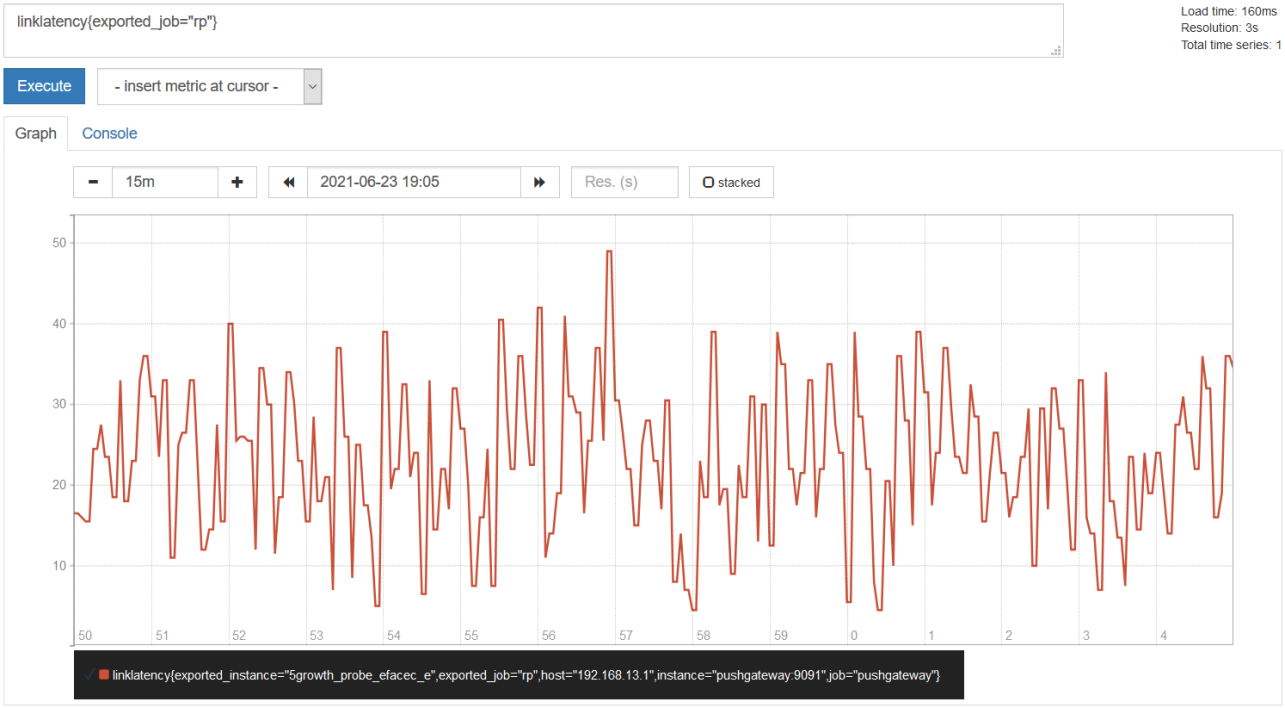

FIGURE 57: 5G MONITORING PLATFORM – U2 LATENCY IN A 15 MINUTES PERIOD

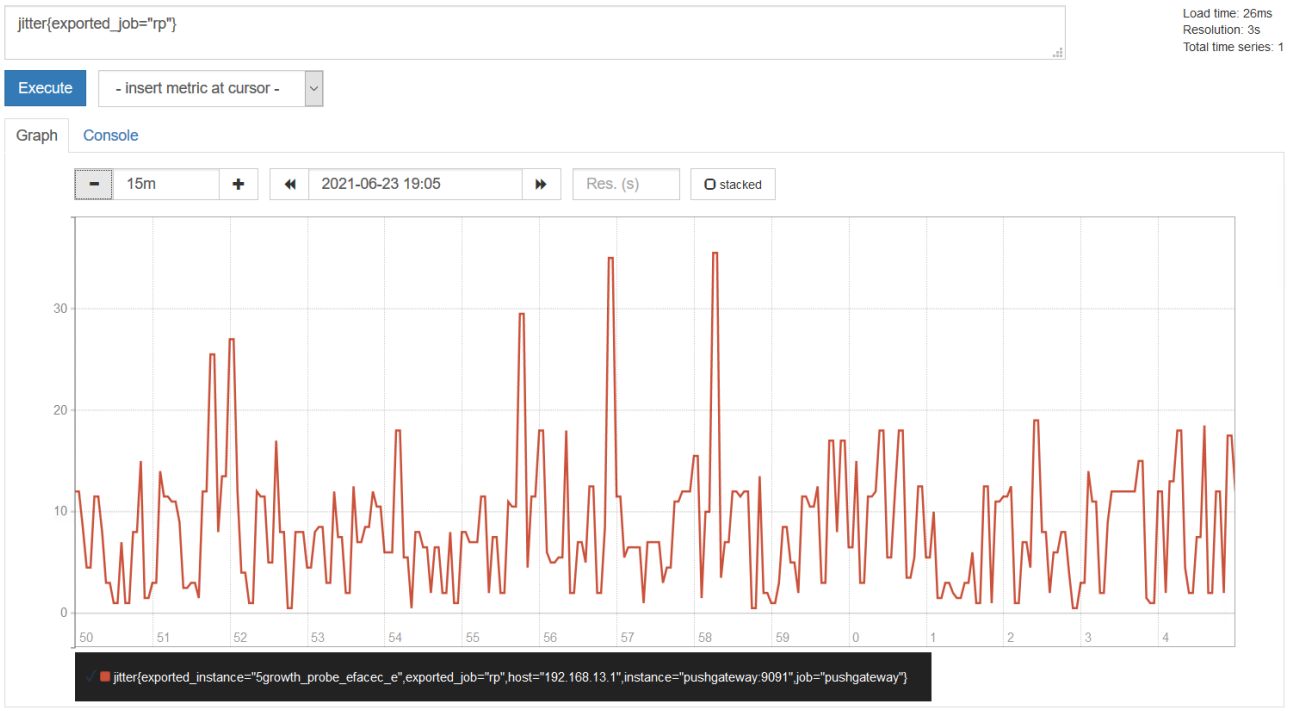

FIGURE 58: 5G MONITORING PLATFORM – UC2 JITTER IN A 15 MINUTES PERIOD

5. Conclusions

This document reports the increased maturity of the 5Growth technical solutions, initiated in D4.2, providing a validation of the new features and functionalities available to verticals. This macroscopic objective is attained by executing a broader validation campaign to verify and demonstrate the fulfilment of the vertical requirements. D4.3 provides evidence of the work carried out since the previous validation report, namely:

- A data infrastructure for collecting, integrating, and making available to different consumers the data produced by the validation experiments.
- A 5Growth experiment catalogue, incorporating a specific information model (IM) for building 5Growth experiment descriptors, as well as an update on the tools supporting the pilot validation campaigns.
- The identification of pilot technical requirements, KPIs and measurement procedures guiding pilot validation campaign.
- The integration process that makes available to the 5Growth Monitoring Platform the collected monitoring data from experiments in the context of 5G EVE and 5G VINNI facilities.
- The results and critical analysis of each pilot second validation campaign as well as description of setups and infrastructure on which it was performed.

The structure of D4.2 provided the general pattern for further contents of WP4 deliverables for this and the coming, last cycle planned for the project. These deliverables are structured around the same idea of combining reports on the evolution of methodology and tooling, and the analyses of verification results evolving according to the scheme depicted in the figure below.

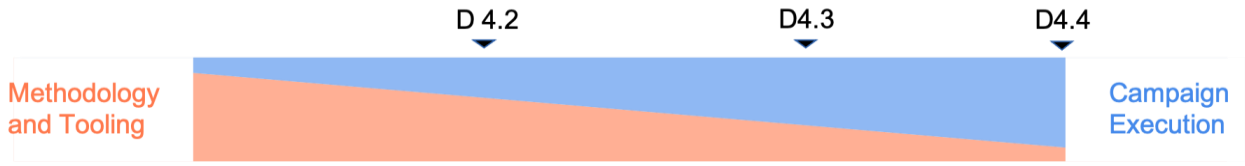

FIGURE 59: FORESEEN EVOLUTION OF WP4 DELIVERABLE CONTENT

With the amount of information focused on methodology and tooling decreasing, while the one reporting results and analyses from verification increases as the project matures.

6. References

[1] ETSI ISG CIM standard: GS CIM 009 - V1.4.2 - Context Information Management (CIM); NGSI-LD API (2021

[2] Apache NiFi. Available at: https://nifi.apache.org/.

[3] Apache SparkTM - Unified Analytics Engine for Big Data. Available at: https://spark.apache.org/.

[4] Apache Flink: Stateful Computations over Data Streams. Available at: https://flink.apache.org/.

[5] Kafka Streams. Available at: https://kafka.apache.org/documentation/streams/.

[6] Apache Kafka. Available at: https://kafka.apache.org/.

[7] FastAPI. Available at: https://fastapi.tiangolo.com/.

[8] ScorpioBroker: NGSI-LD compliant context broker named Scorpio. Developed by NEC Laboratories Europe and NEC Technologies India. Available at: https://github.com/ScorpioBroker/ScorpioBroker.

[9] Prometheus – Monitoring system & time series database.". Available: https://prometheus.io/.

[10] Apache AvroTM 1.10.2 Documentation. Available at: https://avro.apache.org/docs/current/.

[11] Protocol Buffers | Google Developers. Available at: https://developers.google.com/protocol-buffers.

[12] gRPC Network Management Interface (gNMI) specification.". Available: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md

[13] Containerized EOS (cEOS) - Software Controlled Container Networking - Arista. Available at: https://www.arista.com/en/products/software-controlled-container-networking.

[14] gNMIc. Available at: https://gnmic.kmrd.dev/.

[15] 5Growth "Final Design and Evaluation of the innovations of the 5G End-to-End Service Platform." Deliverable 2.3, May 2021.

[16] 5G-TRANSFORMER, D3.1, Definition of vertical service descriptors and SO NBI, March 2018.

[17] Ref to 5G EVE D3.4: Second implementation of the interworking reference model. Available here: https://doi.org/10.5281/zenodo.3946323

[18] Minimum requirements related to technical performance for IMT-2020 radio interfaces: https://www.itu.int/pub/R-REP-M.2410

[19] 5G-EVE, D4.1, Experimentation Tools and VNF Repository. October 2019

[20] "Latency in Optical Fiber Systems", available online at: https://www.commscope.com/globalassets/digizuite/2799-latency-in-optical-fiber-systems-wp-111432-en.pdf?r=1