5G European Validation platform for Extensive trials

Deliverable D5.1
Disjoint testing and validation tools

Project Details

Call	H2020-ICT-17-2018
Type of Action	RIA
Project start date	01/07/2018
Duration	36 months
GA No	815074

Deliverable Details

Deliverable WP:	WP5
Deliverable Task:	Task T5.3
Deliverable Identifier:	5G_EVE_D5.1
Deliverable Title:	Disjoint testing and validation tools
Editor(s):	Christos Ntogkas, Evangelos Kosmatos
Author(s):	Christos Ntogkas, Juan Rodriguez Martinez, Jaime Garcia-Reinoso, Evangelos Kosmatos, Claudio Casetti, Gino Carrozzo, Elian Kraja, Leonardo Agueci, Kostas Trichias, Bougioukos Marios, Vrakas Nikos, Laurent Roullet, Arturo Azcorra Saloña, Carlos Guimaraes, Pablo Serrano Yáñez-Mingot, Panagiotis Demestichas, Despoina Meridou, Orestis Zekai, Vera Stavroulaki, Nelly Giannopoulou, Ioannis Belikaidis.
Reviewer(s):	Juan Rodriguez Martinez, Evangelos Kosmatos
Contractual Date of Delivery:	30/4/2019
Submission Date:	03/5/2019
Dissemination Level:	PU
Status:	1.0
Version:	Final
File Name:	5G_EVE_D5.1

Deliverable History

Version	**Date**	**Modification**	**Modified by**
V0.01	*31/21/2018*	*Template*	*Kostas Trichias*
V0.02	*5/4/2019*	*TID, UC3M contributions*	*Juan Rodriguez Martinez, Jaime Garcia-Reinoso*
V0.03	*10/4/2019*	*WINGS contributions*	*Christos Ntogkas, Evangelos Kosmatos*
V0.04	*15/4/2019*	*CNIT contributions*	*Claudio Casetti*
V0.05	*15/4/2019*	*NXW contributions*	*Elian Kraja, Leonardo Agueci*
V0.06	*16/4/2019*	*TID, UC3M updates*	*Juan Rodriguez Martinez, Jaime Garcia-Reinoso*
V0.07	*17/4/2019*	*WINGS updates*	*Christos Ntogkas, Evangelos Kosmatos*
V0.08	*17/4/2019*	*Editorial updates*	*Christos Ntogkas, Evangelos Kosmatos*
V0.09	*18/4/2019*	*NOK-GR contributions*	*Bougioukos Marios, Vrakas Nikos*
V0.10	*22/4/2019*	*Reviewed by WINGS*	*Evangelos Kosmatos*
V0.11	*23/4/2019*	*Reviewed by NXW and TID*	*Gino Carrozzo, Juan Rodriguez Martinez*
V0.12	*25/4/2019*	*Comments addressed*	*Evangelos Kosmatos*
V0.13	*30/4/2019*	*Comments and editing*	*Juan Rodriguez Martinez*
V0.14	*30/4/2019*	*NOK-FR contributions*	*Laurent Roullet*
V0.15	*1/5/2019*	*Final editing*	*Evangelos Kosmatos*

Table of Contents

List of Acronyms and Abbreviations

Acronym	*Meaning*
5G PPP	5G Infrastructure Public Private Partnership
AGV	Autonomous Guided Vehicles
API	Application Programming Interface
CSAR	Cloud Service Archive
DF	Deployment Flavours
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DNS	Domain Name System
EAF	Experiment Automation Framework
EDE	Experiment Definition and Execution
ELK	Elasticsearch, Logstash and Kibana
eNB	Evolved Node B
ExpD	Experiment Descriptors
FDD	Frequency Division Duplex
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IL	Instantiation Levels
IWL	InterWorking Layer
KPI	Key Performance Indicator
LTE	Long Term Evolution
MANO	MANagement and Orchestration
MDI	Media Delivery Index
MEC	Multi-access Edge Computing
MIMO	Multiple Input Multiple Output
MOS	Mean Opinion Score
NFVO	NFV Orchestrator
NR	New Radio
NSD	Network Service Descriptors
OSM	Open Source MANO
OVS	Open Virtual Switch
PLC	Programmable Logic Controller
PPPoE	Point-to-Point Protocol over Ethernet
QoE	Quality of Experience

QoS	Quality of Service
RAN	Radio Access Network
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Networking
SDR	Software Defined Radio
SLA	Service-Level Agreement
SUT	Service Under Test
TC	Traffic Control
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TEMS	Test Mobile System
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	VNF package descriptions

List of Figures

List of Tables

Executive Summary

This document contains the description of a preliminary implementation of WP5 disjoint tools supporting testing and validation of experiments at the different 5G EVE sites. These tools are the core delivery target of Deliverable D5.1. They are defined as “disjoint” since they are not integrated among them, and are not integrated with the rest of the 5G EVE layers (as they have not been delivered yet). However, the tools are important since they represent functionalities which will be offered by the final 5G EVE Platform, and even in this preliminary version, they will be able to help in the support of the first phases of experimentation.

The document begins with a high-level description of the first version of the Model-based Testing Methodology to be adopted in 5G EVE. In that sense, first need is to understand the experiments that verticals will be willing to run on top of 5G EVE infrastructure. It is not feasible to build from scratch a platform ready to plug and play any experiment; therefore we have concentrated on the project Verticals to start identifying testing requirements.

For that purpose, we have created a simple template for Verticals to describe their test plans. In this template, they can describe with little effort items like their objectives, the applications they are providing, the KPIs they would like to measure, and also the list of tests they actually do to validate their applications (on any scenario, not necessarily 5G).

Such template permits us evaluating the feasibility of the proposal and, if passed, building a more detailed (or low-level) test plan, where we include all the technical elements that will permit the test execution over a 5G network (some of which can be even hidden to the Verticals): the underlay infrastructure, the topology, the required tools, the measurement procedures, etc.

Those templates are the main tools the project has identified for the Test Design and Preparation phases. The document also includes an example of Vertical template response, provided by ASTI for their use case. Beyond understanding the requirements, initial responses are also very useful to improve the templates, making it easier for future Verticals to interact with 5G EVE.

Regarding the testing tools developed in this phase of the project, the main features consist of:

- Being able to automate the execution of experiments, mainly by changing the external conditions (background traffic, delay, etc.). This is done via scripts implemented in Robot Framework, and launched via Jenkins, which is the execution engine.
- Being able to capture KPI metrics from the experiment, and to make them available for the analytics/monitoring modules (using a Kafka bus).
- Being able to validate the metrics against some thresholds, determining the correct operation of the experiment (for which Robot Framework is also used), and also to visualize them (using Grafana) so that they can be presented to the experimenters.

The list of delivered tools is extended in order to provide a full overview of the testing capabilities at the various sites, for which the list of available testing/validation tools in each 5G EVE site is also reported. Some of these tools are part of the common ones being delivered, but most of them are new, specific in many cases to the Use Cases that need to run over each site. In that sense, even some commercial tools are required for very deep testing and analysis.

The delivered tools are available on the project site, in the following link: https://www.5g-eve.eu/end-to-end-facility/

Finally, initial testing and validation results are presented from the use cases by ASTI (Industry 4.0), Utilities and Smart City, with reference to initial mock-ups of the testing GUIs. These examples demonstrate the capabilities of the proposed tools, and also an initial approach on how they can be combined with additional reporting modules, like those also offered by Jenkins.

1 Introduction

The 5G EVE project aims at providing among others, a testing and validation framework for the Verticals to create their tests and validate their services. In this direction, as an intermediate step, a set of disjoint testing and validation tools have been developed and made available to the site owners. The concept of "disjoint" implies that the tools:

- Are not yet fully integrated among them
- Are not yet integrated with the rest of the 5G EVE layers

The delivered tools are available here: https://www.5g-eve.eu/end-to-end-facility/

There are two types of tools: those that are specific to the different sites, and which are not necessarily shared since they focus on the use cases to be deployed at each site, and those that are more generic, and thus reusable across the sites to resemble features that will be available when the 5G EVE Platform is finalized and fully integrated.

1.1 Structure of the document

The main structure of this deliverable can be summarized as follows:

- Section 2 presents a high-level description of the first version of the Model-based Testing Methodology for 5G EVE, with special focus on the first stages: test design and test preparation. The final version of the methodology will be reported in next deliverable D5.2 planned for end of Q4-2019.
- Section 3 presents the high-level design, and implementation aspects, of the first testing and validation tools which WP5 has made available to the sites, and which are the main delivery target of D5.1. In addition, the already available testing tools of the sites are reported. Finally, initial testing and validation results are presented, obtained from the application of the provided tools into selected use cases.
- Section 4 reports conclusions of the document and gives summary directions on future work.

2 Methodology for 5G testing and validation

Multiple tools exist in state of the art for testing and validation purposes, many of which in use from various 5G PPP projects of Phase 1 and Phase 2. Under such enormous offer, the choice of 5G EVE consortium is to follow a Model-based Testing Methodology to represent the desired behaviour of a System Under Test (SUT), the related testing strategies and test environment.

This section includes a high-level description of the first version of the Model-based Testing Methodology for 5G EVE, being specified in Task 5.1. Although the final outcome of T5.1 is to be reported in Deliverable D5.2 by Q4-2019, the current status of specification of this methodology in 5G EVE is here reported to help readers to correctly position the 5G EVE proposed tools in the experiment execution workflows that are being designed.

The main global service provided by the 5G EVE platform for external experimenters is actually the Testing and Validation capability. Independently of low-level conditions, requirements, etc., the high-level workflow for a validation process defined in WP1, consists of the four phases presented in the following Figure 1:

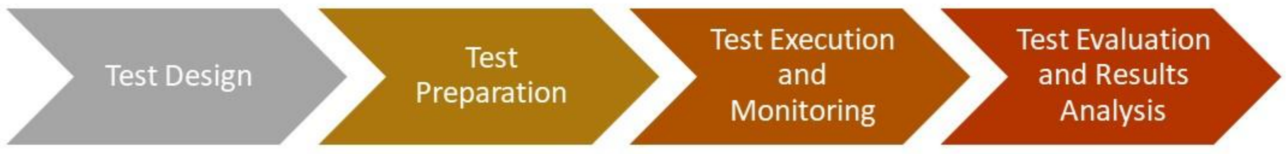

Test Design
Test Preparation
Test Execution and Monitoring
Test Evaluation and Results Analysis

Figure 1: 5G EVE high-level validation process workflow

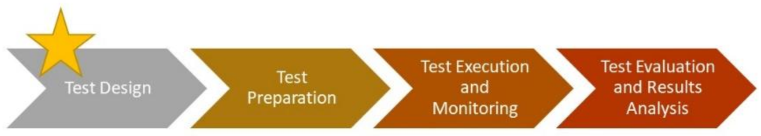
During the Test Design phase, the direct action and inputs from experimenters are critical. In this phase, the experimentation goals and expected results are jointly defined, together with the test external conditions and assumptions. This, together with the information about what components the Vertical is bringing to the experiment, generates a test case specification to which the 5G EVE consortium can execute a feasibility analysis.

If passed, the test case specification which defines the experiments at the experimenter-level must be translated into a more technical and execution-oriented language. In other words, a meaningful test plan is to be generated, including not only the inputs from the experimenters, but also the testing procedures, the test configurations, the measurable KPIs (vertical- and network-related), the required supporting infrastructure, etc. This process is transparent for the experimenter, except for the fact that they need to provide to the platform the appropriate versions of their own components, which may need some adaptation. For example, if a Vertical requires to test a VNF on the 5G EVE platform, he will have to provide the correct image and descriptors so it can actually be deployed. The final outcome of this phase is a planning handshake, which will determine the time frame when the experimenter will be able to conduct its tests.

During the execution phase, all the experiment components are instantiated, all the configurations are put in place and, in summary, the infrastructure is effectively adapted to the requirements of the test plan. Only after every component is ready and correctly configured will the experiment start. In line with that, the experiment and the network conditions will be continuously monitored, with two main purposes: to detect malfunctioning as soon as possible, so the execution can be stopped and resources are not wasted in vain; and to be able to correlate, during the analysis phase, the final results with the conditions at each moment.

Finally, results are collected and analysed, and a final report is generated. There are many levels of analysis, depending on many occasions on the KPIs which are meaningful for each specific Vertical. Network KPIs will be measured by default by the 5G EVE platform, in order to validate the results against the 5G PPP KPIs and as a way to ensure that the obtained results are valid (i.e. obtained under correct network conditions). Those Vertical experiments for which these Network KPIs are enough for their service validation will not need additional efforts to define further tests and post-execution analysis. The same can be said for those who can infer a direct relationship between Vertical-specific and Network-related KPIs. However, for those Verticals interested in KPIs for which it is not possible to establish a clear relationship with Network KPIs, or for those to which this relationship is unknown, the project will still provide mechanisms for defining and exporting Vertical-defined KPIs through the validation tools. As such, an analysis can still be performed based on determined thresholds.

2.1 Vertical test plan


Test Design
Test Preparation
Test Execution and Monitoring
Test Evaluation and Results Analysis

As commented above, the main goal of the Test Design phase is to understand what the Vertical needs concerning experimentation at 5G EVE are. Although some tools exist related to test specification, normally these are more focused on making the life of the test design technician easier (e.g. time management applications, equipment catalogues, etc.) rather than on permitting a third party to express particular needs, like it is required in the project. Furthermore, the process is even more complex if we consider that not all Verticals have the technical knowledge related to 5G networks, MEC, SDN, MANO, etc., as to be able to prepare a test plan on their own. Therefore, this is (still) a process very much based on human interaction, discussions, clarifications, etc., similar to the requirements specification phase on engineering projects, where adequate interaction with the client is key to the project success.

In that sense, the 5G EVE consortium has prepared an experiment description template, as a first input to be gathered from Verticals willing to experiment using the project infrastructure. One important fact in the interaction between Verticals and Telcos (or Manufacturers) is that the “language” is not always the same. Telcos tend to focus on network parameters, conditions, SLAs, etc., while the implications of these concepts are not always fully understood by Verticals. Therefore, a lot of effort has been put in the template to make it understandable for experimenters, hiding the network related components as much as possible.

The template has been shared with the Verticals present in 5G EVE as well, to learn about the correctness of its specification. This is extremely important in the future, as 5G EVE will need to host external experiments as well as those from project partners. Valuable suggestions have been received during the process, ending in a template defined as in section 2.1.1.

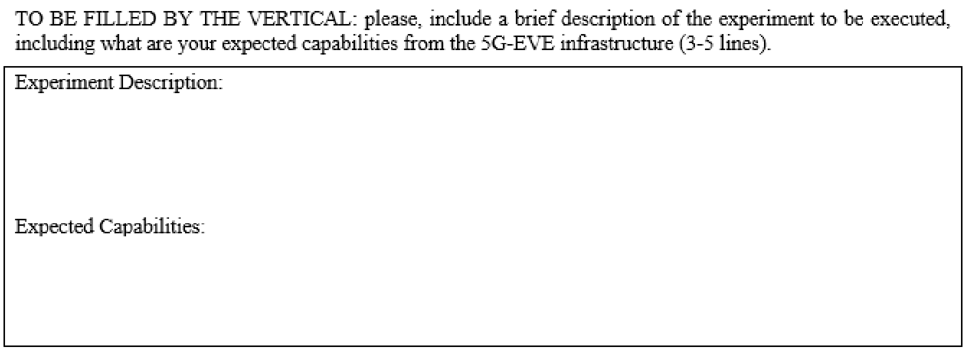
The current template is a Word document, which can be easily shared with Verticals. Improved mechanisms (using online survey mechanisms, for example) may be implemented in the future if needed.

2.1.1 Experiment template for Verticals

Two sections in the current template need to be filled by Verticals. Both sections count with specific instructions, dealing also with issues like the naming criteria.

2.1.1.1 Experiment Description

In the first section, the Vertical is asked to include a brief description of the experiment, and to outline why 5G EVE is adequate for such experiment (in other words, what capabilities the experimenter is seeking from us).



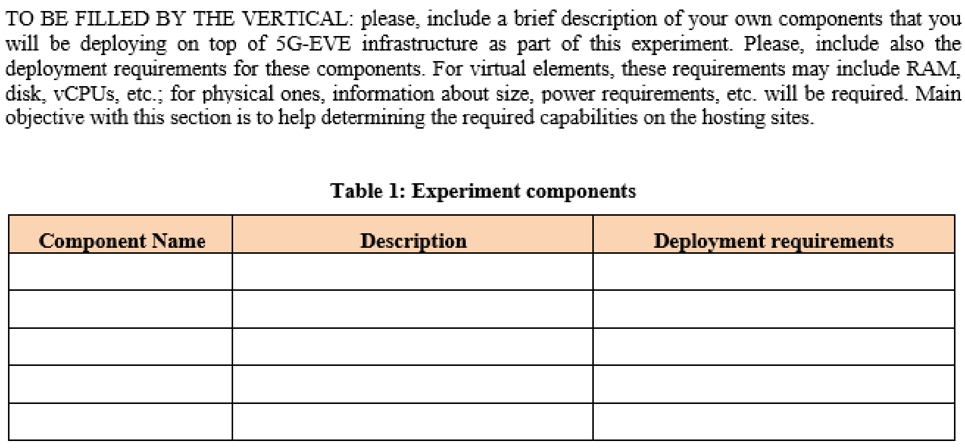
TO BE FILLED BY THE VERTICAL: please, include a brief description of the experiment to be executed, including what are your expected capabilities from the 5G-EVE infrastructure (3-5 lines).

Experiment Description:

Expected Capabilities:

Figure 2: Template for Verticals – Experiment description

The Vertical is also asked to provide a list of the components that it will be bringing into 5G EVE as part of the experiment, together with their deployment requirements and how they interact with each other. This is important, on the one hand to help determining hosting capabilities for the different sites, and on the other for 5G EVE to understand the required connectivity among components.

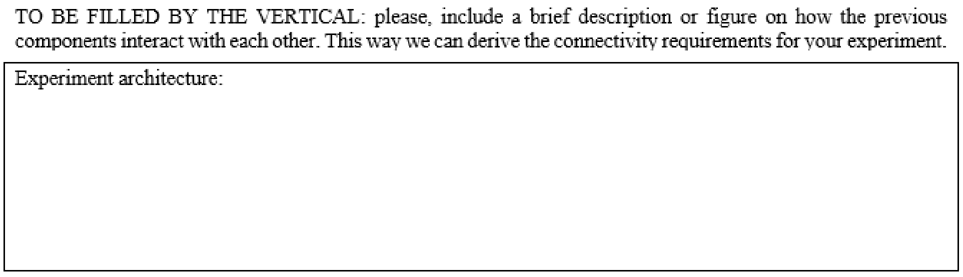


TO BE FILLED BY THE VERTICAL: please, include a brief description of your own components that you will be deploying on top of 5G-EVE infrastructure as part of this experiment. Please, include also the deployment requirements for these components. For virtual elements, these requirements may include RAM, disk, vCPUs, etc.; for physical ones, information about size, power requirements, etc. will be required. Main objective with this section is to help determining the required capabilities on the hosting sites.

Table 1: Experiment components

Component Name	Description	Deployment requirements

Figure 3: Template for Verticals – Experiment components

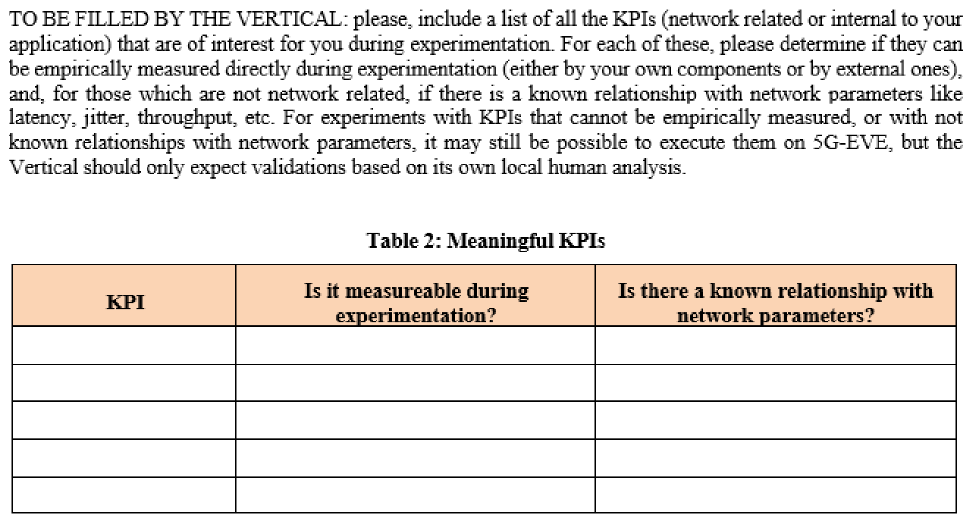


TO BE FILLED BY THE VERTICAL: please, include a brief description or figure on how the previous components interact with each other. This way we can derive the connectivity requirements for your experiment.

Experiment architecture:

Figure 4: Template for Verticals – Experiment architecture

Finally, the Vertical is asked to list the KPIs which are meaningful for them, if these KPIs can be measured during experimentation, and for those which are not network-related, if there is a known relationship with network parameters on which we can leverage for the analysis of results.



TO BE FILLED BY THE VERTICAL: please, include a list of all the KPIs (network related or internal to your application) that are of interest for you during experimentation. For each of these, please determine if they can be empirically measured directly during experimentation (either by your own components or by external ones), and, for those which are not network related, if there is a known relationship with network parameters like latency, jitter, throughput, etc. For experiments with KPIs that cannot be empirically measured, or with not known relationships with network parameters, it may still be possible to execute them on 5G-EVE, but the Vertical should only expect validations based on its own local human analysis.

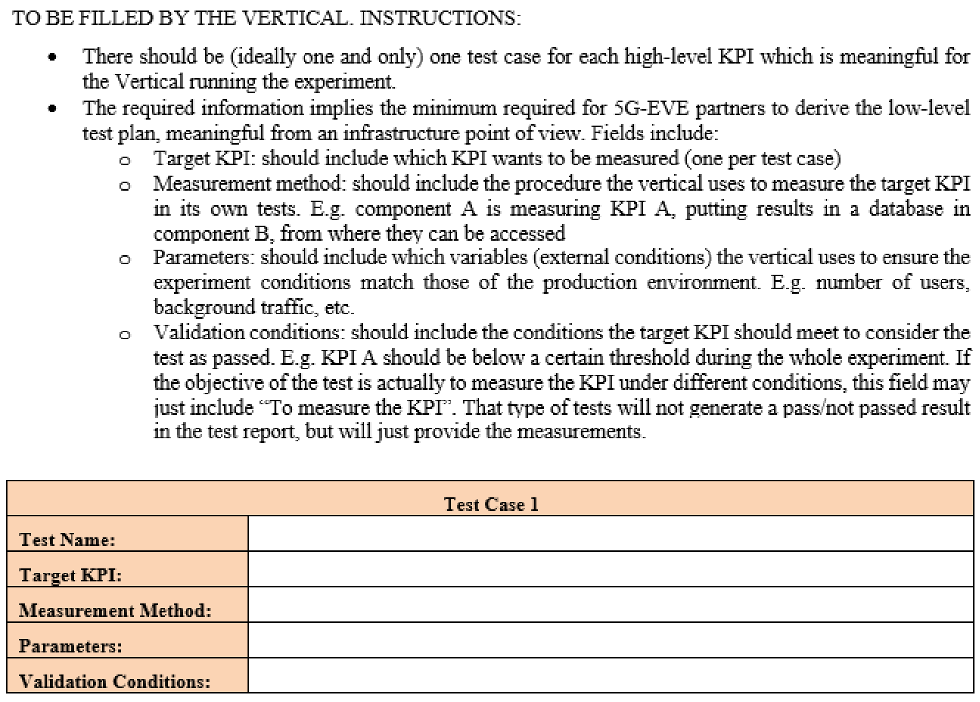
Table 2: Meaningful KPIs

KPI	Is it measureable during experimentation?	Is there a known relationship with network parameters?

Figure 5: Template for Verticals – Experiment KPIs

2.1.1.2 Test Plan

The second section of the template includes a list of test cases, ideally one per target KPI. Beyond the test name, and which is that target KPI, Verticals are asked about the measurement method, about the test variables (external conditions) they would like to have available to ensure that the test is executed under appropriate circumstances, and about what the validation conditions are (e.g. threshold for a specific variable).



TO BE FILLED BY THE VERTICAL. INSTRUCTIONS:

- There should be (ideally one and only) one test case for each high-level KPI which is meaningful for the Vertical running the experiment.
- The required information implies the minimum required for 5G-EVE partners to derive the low-level test plan, meaningful from an infrastructure point of view. Fields include:
  - Target KPI: should include which KPI wants to be measured (one per test case)
  - Measurement method: should include the procedure the vertical uses to measure the target KPI in its own tests. E.g. component A is measuring KPI A, putting results in a database in component B, from where they can be accessed
  - Parameters: should include which variables (external conditions) the vertical uses to ensure the experiment conditions match those of the production environment. E.g. number of users, background traffic, etc.
  - Validation conditions: should include the conditions the target KPI should meet to consider the test as passed. E.g. KPI A should be below a certain threshold during the whole experiment. If the objective of the test is actually to measure the KPI under different conditions, this field may just include "To measure the KPI". That type of tests will not generate a pass/not passed result in the test report, but will just provide the measurements.

Test Case 1	
Test Name:	
Target KPI:	
Measurement Method:	
Parameters:	
Validation Conditions:	

Figure 6: Template for Verticals – Experiment Test Cases

Verticals can include as many test cases as required.

2.1.2 Example of Vertical experiment template for ASTI use case

In order to validate the template for the Verticals presented in section 2.1.1, we have worked with ASTI, one of the Verticals involved in the 5G EVE project, to discuss their planned validation experiments. To guide the work, we used the aforementioned Template for Verticals and asked ASTI to fill it (in the form of a questionnaire), in order to identify the critical aspects of the tests and the expected validation priorities of the vertical. In the next subsections, we present their answers to all these questions.

2.1.2.1 Experiment description by ASTI

With respect to the questions about the experiment description and the expected capabilities, already presented in Figure 2, this is the answer provided by ASTI:

Experiment Description:

The experiment will test the virtualization of the control algorithms of AGVs (Autonomous Guided Vehicles). In the current state of the art the AGVs have a PLC in charge of governing the internal control loop, what is collecting the information of the guiding sensors, taking the appropriate control decisions and generating the necessary signals to regulate the speed of the motors. During the experiment this internal control loop will be relocated out of the AGV thanks to the 5G communication technologies.

Expected Capabilities:

A high performance and reliable mobile network connecting the AGVs to the platform with the virtual machines is required. The most challenging requirements to ensure the deployment of the solution in the future factories are the high reliability and the low latency and jitter.

As commented by ASTI, the main capabilities of a 5G network required by their use case is high reliability, low latency and jitter for the communication between the AGVs and the control elements placed in the edge. To have a complete picture about all elements involved in these experiments, the following is the answer from ASTI about the experiment components already presented in Figure 3.

Table 1: ASTI Experiment components

Component Name	Description	Deployment requirements
AGV	*AGV where the slave PLC is running*	*220V to power a charging station* *220V to power a traffic station* *Test zone of 9mx4m*
Slave PLC	*PLC included in the AGV*	*Client device to connect to 5G infrastructure. 24Vdc powered or with some dc/dc converter (for example 24/5 if the client device is 5V powered).* *Static IP*
Master PLC	*Virtual PLC embedded in a Virtual Machine*	*Virtual machine to run Windows 7 (Service Pack 1 or higher) / 8 / 10 (32/64 Bit) + suitable PC hardware for the corresponding Windows platform*
Delay configurator		*Device to adjust the network delay*
Traffic generator		*Used to inject traffic into the network, simulate virtual AGVs and other external traffic*

This information, together with the experiment architecture presented below, constitute the main elements that have to be considered to define the environment to run the correspondent tests.

Experiment architecture:

The AGV will have a Slave PLC on board. This PLC will collect the information from the sensors and physical inputs, this will be sent to the Virtual PLC. The Virtual PLC will process all this information, then it will take the appropriate control decisions and it will generate the right signals to control the motors of the AGV. This control signals will be sent by 5G communication network to the Slave PLC, which will process them translating to real physical signals to command the motors.

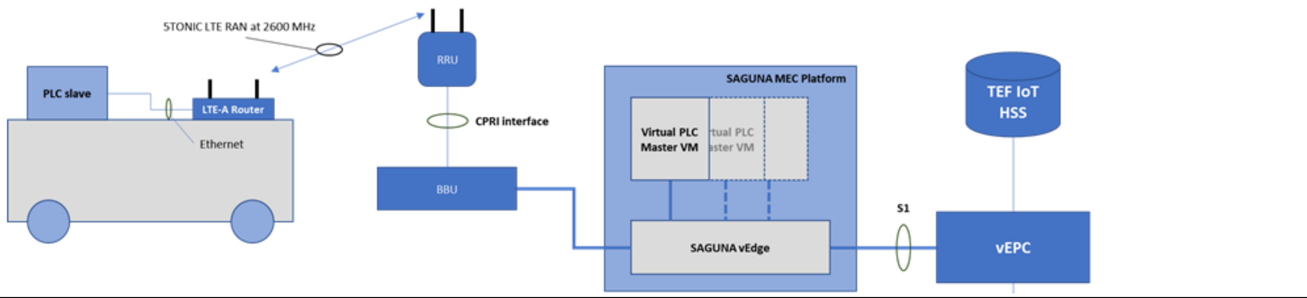


Figure 7: Experiment architecture

At the end of the day, the Key Performance Indicators (KPIs) for the vertical (ASTI in this particular example) constitute the core of all experiments, which have to be carefully defined to check if they are satisfied in different conditions. These are the KPIs provided by ASTI:

Table 2: ASTI Vertical KPIs

Vertical KPI	Is it measurable during experimentation?	Is there a known relationship with network parameters?
Navigation level	*Yes by the AGV*	*If the network delay is high the navigation deteriorates*
Consumption	*Yes by the AGV*	*If the network delay is high the consumption increases*
Time to lose the guide	*Yes by the AGV*	*If the network delay is high the time decreases*
Delay	*Yes*	*Network delay*
Number of Virtual AGVs supported	*Yes*	*Less delay and more available bandwidth will increase this number*

2.1.2.2 Test plan by ASTI

After presenting the core components and vertical KPIs involved in the ASTI experiments, this vertical has proposed the following four test cases:

Test Case 1	
Test Name:	*Navigation level at different network delays*
Target Vertical KPI:	*Navigation level*
Measurement Method:	*The navigation level will be measured by the AGV*
Parameters:	*The AGVs will run 10 cycles. Without external traffic. The delay will be fixed at 10ms, 15ms, 20ms, … until the maximum acceptable delay*

Expected Results:	*Determine how the navigation level is affected by the network delay*

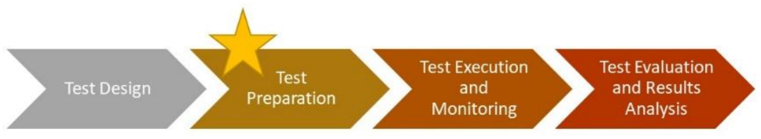
Test Case 2	
Test Name:	*Maximum supported delay*
Target Vertical KPI:	*Time to lose the guide, Delay*
Measurement Method:	*One/both AGVs stop when the guide is lost*
Parameters:	*Without external traffic. The delay is increased until one AGV loses the guide*
Expected Results:	*Determine the maximum supported network delay*

Test Case 3	
Test Name:	*Acceptable supported delay*
Target Vertical KPI:	*Navigation level, Delay*
Measurement Method:	*Navigation level of one AGV is below an acceptable level*
Parameters:	*Without external traffic. The delay is increased until the navigation level is not acceptable*
Expected Results:	*Determine the acceptable supported network delay*

Test Case 4	
Test Name:	*Consumption at different network delays*
Target Vertical KPI:	*Consumption, delay*
Measurement Method:	*The consumption will be measured by the AGV*
Parameters:	*The AGVs will run 10 cycles. Without external traffic. The delay will be fixed at 10ms, 15ms, 20ms, ... until the maximum acceptable delay*
Expected Results:	*Determine how the consumption is affected by the network delay*

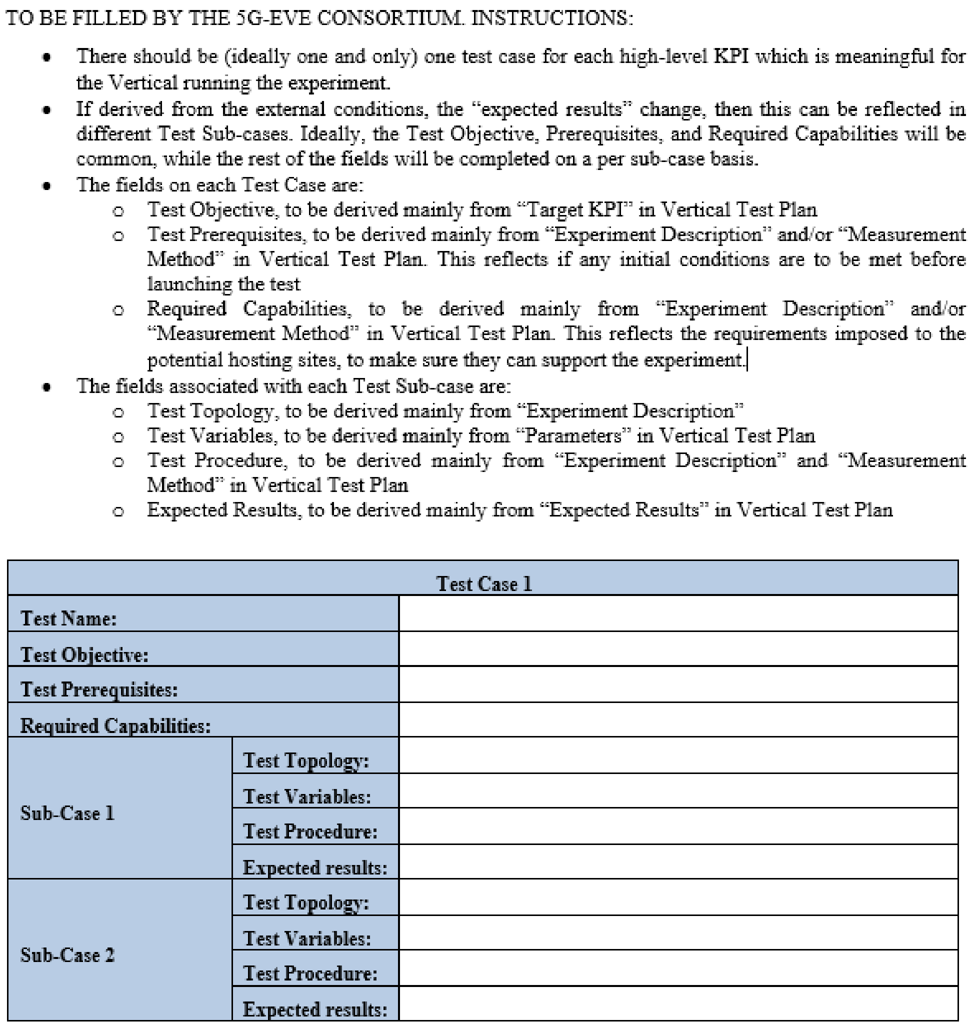
These test cases impose some requirements to the 5G network infrastructure in terms of capabilities and performance, both at the radio and the edge part, but they also require a set of testing tools to provide an environment which emulates the same conditions this vertical could find in a real deployment. These testing tools will be presented in Section 3.

2.2 Low level test plan


Test Design
Test Preparation
Test Execution and Monitoring
Test Evaluation and Results Analysis

Based on the Vertical inputs on the previous questionnaire, the consortium can then infer a lower level test plan, valid to start the interactions with the different sites and Site Managers. For that purpose, a second template (Figure 8) has been generated, including all the information that typically populates test plans. Beyond fields like the test pre-requisites, topology, procedure, and so on, the field on “Required Capabilities” is very important since it will permit matching the needs of the experiment with the features offered by the different sites. In case there are limitations, it will be possible to launch an experiment only at specific project sites, situation that will be reflected in the Portal for the experimenter to be aware of. This information may also be used as an input for

WP2: in case sites have the chance to increment their capabilities, they will be able to prioritize features based on experiments needs.



TO BE FILLED BY THE 5G-EVE CONSORTIUM. INSTRUCTIONS:

- There should be (ideally one and only) one test case for each high-level KPI which is meaningful for the Vertical running the experiment.
- If derived from the external conditions, the "expected results" change, then this can be reflected in different Test Sub-cases. Ideally, the Test Objective, Prerequisites, and Required Capabilities will be common, while the rest of the fields will be completed on a per sub-case basis.
- The fields on each Test Case are:
  - Test Objective, to be derived mainly from "Target KPI" in Vertical Test Plan
  - Test Prerequisites, to be derived mainly from "Experiment Description" and/or "Measurement Method" in Vertical Test Plan. This reflects if any initial conditions are to be met before launching the test
  - Required Capabilities, to be derived mainly from "Experiment Description" and/or "Measurement Method" in Vertical Test Plan. This reflects the requirements imposed to the potential hosting sites, to make sure they can support the experiment.
- The fields associated with each Test Sub-case are:
  - Test Topology, to be derived mainly from "Experiment Description"
  - Test Variables, to be derived mainly from "Parameters" in Vertical Test Plan
  - Test Procedure, to be derived mainly from "Experiment Description" and "Measurement Method" in Vertical Test Plan
  - Expected Results, to be derived mainly from "Expected Results" in Vertical Test Plan

Test Case 1		
Test Name:		
Test Objective:		
Test Prerequisites:		
Required Capabilities:		
Sub-Case 1	**Test Topology:**	
	Test Variables:	
	Test Procedure:	
	Expected results:	
Sub-Case 2	**Test Topology:**	
	Test Variables:	
	Test Procedure:	
	Expected results:	

Figure 8: Low Level Test Plan Template

It may happen that, depending on several internal or external conditions, for the same KPI the validation conditions change. For example, a specific VNF may support a maximum number of clients depending on the number of assigned virtual cores. The Vertical providing the VNF may want to deploy three flavours of the VNF, with 1, 2 and 4 cores respectively, and check how they behave depending on the delay. While it may be argued that these are three different use cases, in 5G EVE we will consider that, since the target KPI is the same, it is easier to split them in different sub-cases. The VNF flavours can be reflected based on slightly different test topologies, and each sub-case will have its own validation conditions (e.g. the VNF with 1 core will PASS the test if it serves 1k subscribers, while the VNF with 4 cores will PASS it if it servers 4k subscribers).

In other words, sub-cases represent tests which are essentially the same, measuring the same KPI, but in which the validation threshold is different for whatever cause.

The test cases derived from the low-level test plan will be those which will be available for verticals to select at the Experimentation Portal, very similarly to what "wizards" do in commercial traffic generation tools: these test cases represent very common scenarios, which may be useful for different experimenters, so the intent-based specification is somehow simplified by asking the experimenter to fill-in only the few parameters that govern the scenario.

2.3 Test procedures

This section describes the test execution procedures, that is, the processes that happen after an experiment is instantiated and deployed. How this deployment is done is to be covered in future Deliverables of WP3.

Test procedures vary, of course, depending on the type of test that is being executed. Defining the specifics of every possible test, even only of those which may be meaningful for 5G EVE Verticals, would be useless at this stage. Instead, differences between two complementary types of tests will be analysed. Also, important concepts will be defined, like KPIs, validation parameters and testing variables.

The first type of tests is what will be called "technology benchmarking". The purpose of these tests is to characterize some component, technology, architecture, etc., normally based on network parameters. Many network KPIs, as presented in work from WP1, can be measured as part of a 5G-related benchmarking; in this first Deliverable we will be concentrating on two of the more critical ones: bandwidth and delay. The second type of tests which may be executed on top of 5G EVE infrastructure is the testing of "vertical applications". In this case, experimenters are more interested in what is the behaviour of their application on top of a 5G network, in general, or on top of specific 5G components, in particular.

2.3.1 Definitions

Before moving into the description of procedures, it is important to differentiate between:

- **KPIs**: key performance indicators represent measurable scenario parameters which provide meaningful performance information to the experimenter. KPIs can be of different types (network related or vertical related, for example), and the type of information that they provide can be used for many different purposes. Validation and monitoring are just two of these purposes.
- **Validation parameters**: KPIs used to determine the result (PASS, FAIL) of a determined experiment will be called validation parameters. Together with a threshold, or range, validation parameters create the validation conditions which validation tools will use for their analysis.
- **Testing variables**: these are the parameters that the experimenter will want to modify during the test execution, so that (normally) a relationship between a testing variable and a validation parameter is found.

In section 2.1.2.2, the Vertical test plan provided by ASTI was included. Test Case 1 is defined to understand the behaviour of the AGV (measured in terms of the navigation level) compared with its delay from the Master PLC. In this case, the navigation level itself is a Vertical KPI, while the delay is a Network KPI. Both are measurable (the AGV measures the navigation level) and meaningful parameters for the experimenter, so they comply with the KPI definition. They may not be the only ones, though. For example, the experimenter may be interested, for whatever reason, in the frequency of the 5G link that interconnects the AGV with the Master PLC, or in the remaining available bandwidth.

All KPIs in any Test Case will be monitored, and results will be included in the Test Report.

However, the main objective of this test is precisely to find the relationship between delay and navigation level. The former one, therefore, will be the testing variable, and the latter will be the validation parameter. This relationship in particular will be the main focus of the Test Report.

2.3.2 Technology benchmarking

As mentioned above, this Deliverable concentrates on two of the most important Network KPIs: delay and bandwidth. Procedures to measure both are well-known already, and there are multitude of tools which permit that measurement, so this information will not be included in the text. The only important fact to mention is that it must be clear in each test "what" delay and/or bandwidth is being actually measured (i.e., between which points, under which other conditions, etc.).

Once the measurement end-points (and the external conditions) are defined, the actual test may be executed. Tests are normally of two types: either to determine what the delay and/or bandwidth values are in operational conditions (which may be very useful for monitoring purposes), or to understand how these KPIs are affected by testing variables, which are usually network impairments like network failures, additional delay, jitter, packet loss, etc.

These measurements will not always be done with a specific application in mind, so validation conditions will not always be known (or even important…). In those cases, KPIs will be presented, but not validated. Beyond the graphical representations, results will normally include maximum, minimum and average values.

2.3.3 Vertical applications testing

The main procedure in all vertical applications testing will be **to vary the testing variable until the validation parameter is outside of its expected values**. The main result included in the Test Report, therefore, will be that **the test can be considered as PASSED until a certain value of the testing variable is reached**. In other words, the test FAILS when certain value of the testing variable is reached.

On occasions, the validation parameter may not be directly measurable, but there is a well-established relationship with a measurable network parameter. This should be reflected directly in the low-level test plan, so that when the test is deployed, mechanisms to measure such network parameter are considered.

Validation conditions will not always be defined, or validation parameters may not be measurable for some tests. In this case, the 5G EVE platform will only present the measured KPIs versus the testing variable, without determining the final result of the experiment; this, if desired, shall be done by the experimenter.

Finally, a very interesting practice consists of validating the deployment scenario before the actual testing is executed. Doing a preliminary benchmarking test to ensure that the network is in the appropriate conditions may simplify troubleshooting, in the sense that if the network is already experiencing uncontrolled impairments, it makes no sense to execute any test on top of it.

Having said this, a high-level procedure for Vertical applications testing is depicted in Figure 9.

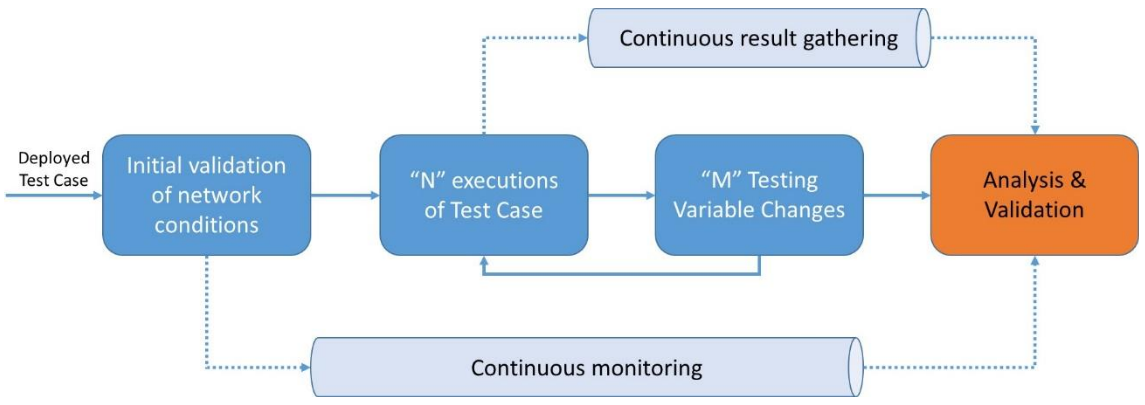

Continuous result gathering
Deployed Test Case
Initial validation of network conditions
"N" executions of Test Case
"M" Testing Variable Changes
Analysis & Validation
Continuous monitoring

Figure 9: Vertical applications testing high-level procedure

3 Disjoint testing and validation tools

This section aims at providing an overview of the disjoint testing and validation tools that were developed during the first months of the project and became available to all site owners and Vertical applications. The objective of the Deliverable is to provide to sites a set of tools for the easy testing of Vertical Services in the first phases of the project, as an addition to the testing tools already available in the sites. In this section, initially a very high-level representation of features that could be supported by the 5G EVE Testing Framework, and that are the basis for the delivered tools, is presented. Then, the implementation aspects of the provided tools are described, together with the already existing testing tools in the sites.

3.1 High level architecture of delivered tools

As stated in previous sections, this Deliverable deals with a set of tools which will permit sites to better support the first phases of testing by the Verticals. They are disjoint tools, which, among other things, implies that they are not integrated with the rest of the 5G EVE layers. In that sense, differences might appear between the high-level architecture of the tools that are being delivered (Figure 10) and the final design of the 5G EVE Experiment Execution and Validation Framework, to be included in future documents from this and other Workpackages.

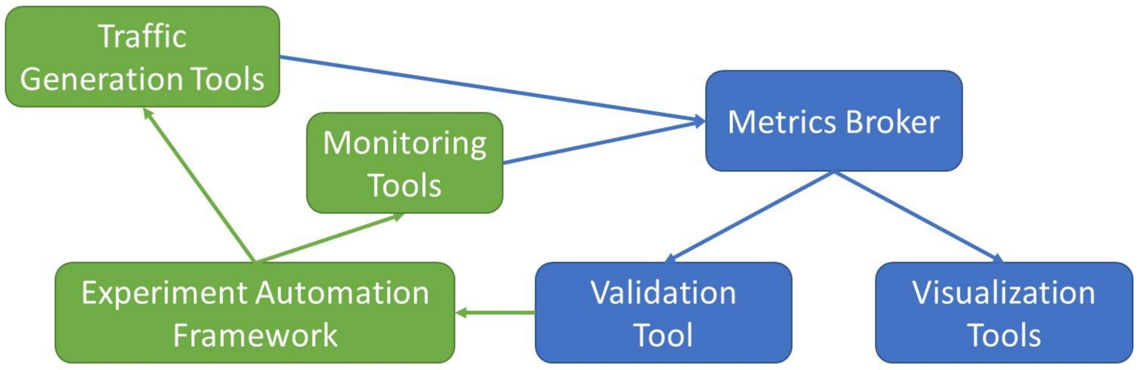


Figure 10: High level description of D5.1 testing tools

The design of the 5G EVE Experiment Execution and Validation Framework will be built around the concept of automation of the experiment execution procedures and KPI evaluation. The objective of the delivered tools is at least to provide a first version of those two concepts.

The Traffic Generation Tools will be able to generate measurement or background traffic, and emulate network impairments like delay or packet loss. These tools will be governed by the Automation Framework, via scripts in this first phase, so that the experiment can be tested under different conditions. Meanwhile, Monitoring Tools will provide to the Metrics Broker both monitoring and experiment result data (in other words, all the KPIs).

Consumers of this data will be both the Validation Tool and the Visualization Tools. The latter will present in various graphs all the KPIs, so the relationships among them can be obtained as a result of the experiment. The Validation Tool, instead, will only consume the data associated with the validation parameters, i.e., those that must be kept under a certain threshold, or between some ranges, to certify that the experiment is operating well. This tool will continuously compare the validation parameters with the thresholds and if the validation condition is met, i.e., if they are outside the correct values, the Validation Tool will be able to stop the test with a command towards the Automation Framework, and report the conditions under which the experiment failed.

3.2 Implementation of delivered tools

In accordance with the aforementioned architecture, the testing tools that are deployed and become available to the sites as part of the current deliverable are following the architecture illustrated in Figure 10. The initial implementation of the different features is shown in Figure 11.

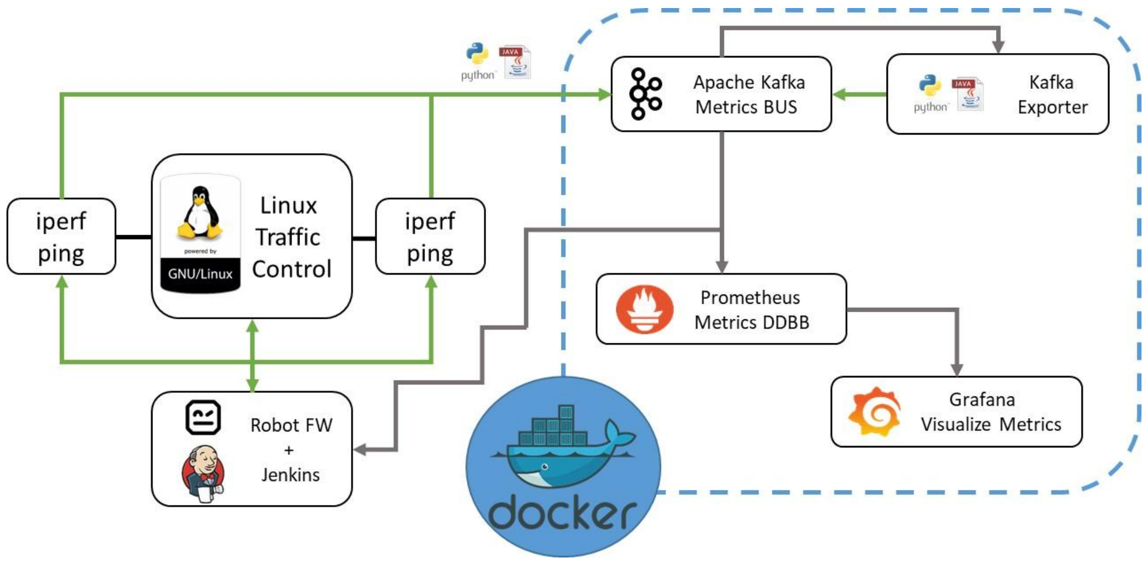


Figure 11: Implementation of D5.1 tools

Traffic Generation Functions are based on three well-known applications, which can be easily deployed at any Virtual Machine of any site:

- "iperf", for the generation of TCP/UDP traffic
- "ping", for delay measurement
- Linux Traffic Control (TC), to emulate additional delay or to generate packet loss

The objective within 5G EVE is to evolve these tools in the future to others which may be more precise; they will suffice however for the first experimentation stages.

The **Experiment Automation Framework** is based on "Robot Framework" (a python-based and extensible keyword-driven test automation framework, used to build the scripts) and "Jenkins" (which is a friendlier execution engine, including also multiple logs for test results). Robot Framework is widely used for end-to-end acceptance testing and acceptance-test-driven development (ATDD). Worth to mention that it is an open source software released under the Apache License 2.0 initially developed at Nokia Networks, nowadays sponsored by Robot Framework Foundation.

The two applications, Robot Framework and Jenkins, have been built in a single Virtual Machine image, including several tests already implemented by the 5G EVE consortium, and the required libraries to automatically run such tests. This way, the Automation Framework can easily be replicated at any site. A configuration guide has also been created, to configure parameters such as the Metrics Broker IP address, which are site dependant.

Automatic tests that are already available include:

- Bandwidth measurement and comparison against a threshold
- Delay measurement and comparison against a threshold
- Delay modification over time, and impact in bandwidth
- Delay modification over time, and comparison of any given validation parameter against a threshold

The **Metrics Broker** has been built using the "Apache Kafka Bus" and a slightly modified version of the "Kakfa Exporter" module from OSM Release FOUR. The selection of Kafka Exporter is derived from the Visualization Tools, which were already integrated with it, but it is not mandatory in the longer term. The modification has been implemented by the project, and it had to do with the fact that it would not be receiving "consensus" data types from a VIM, but any type of measurements; we needed to control the Kafka topics from outside, therefore. It is worth mentioning that we have only extracted this module from OSM; it is not required to install OSM to use this capability.

Other important developments from 5G EVE are the scripts to "create" Kafka metric producers. This is solved for many generally available modules (like OpenStack), but we generated our own scripts both in Python and Java. The objective of such scripts is to facilitate the integration of Vertical KPIs in the validation and visualization phases: Verticals will not need to develop these themselves, but will be able just to reuse them.

The **Validation Tool** in this case is also the "Robot Framework". It can either read metrics back from the prompt when connecting to Traffic Functions Tools (e.g. the delay result of a "ping"), or subscribe to a Kafka topic to receive validation parameters. Initially, "Robot Framework" is built to declare the result of an experiment depending on the execution of all the instructions in the script: if all can be executed, the test is declared as PASSED, independently of the result of these instructions. This has been modified to perform the validation depending on the comparison of the results with a threshold.

Finally, well-known "Prometheus" DDBB and "Grafana" applications forms the **Visualization GUI**, subscribing on specific Kafka topics depending on the specifics of the executed test. A very valuable feature of these tools is that they permit drawing multiple parameters in the same layout, together with the associated thresholds, thus allowing the use of custom visualizations based on the use case needs and specific KPIs, as shown in the example test of Figure 12 regarding the AGVs use case.

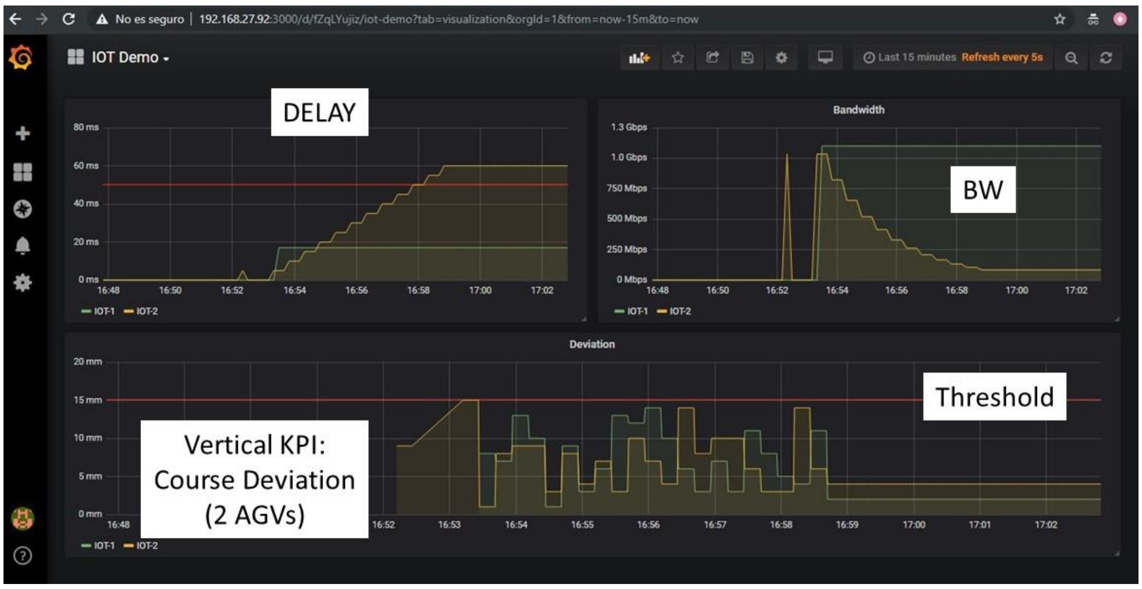

No es seguro | 192.168.27.92:3000/d/fZqLYujiz/iot-demo?tab=visualization&orgId=1&from=now-15m&to=now
IOT Demo
Last 15 minutes Refresh every 5s
DELAY
Bandwidth
BW
Deviation
Threshold
Vertical KPI: Course Deviation (2 AGVs)
IOT-1
IOT-2

Figure 12: Visualization Tools example (AGVs use case)

Another example of the easily modifiable Visualization GUI shown in the example test of Figure 13 regarding the Utilities use case.

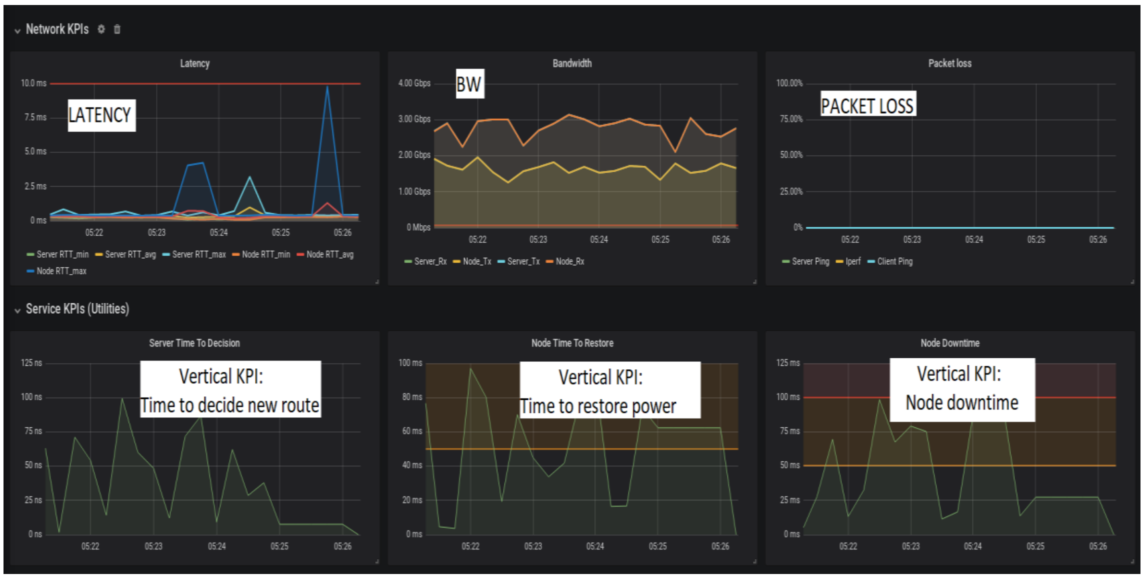


Figure 13: Visualization Tools example (Utilities use case)

The same way that Robot Framework and Jenkins have been built together as a single Virtual Machine, for replication, another VM image has been provided with the four Broker and Visualization Tools (those inside the blue dotted lined in Figure 11). All four modules are instantiated as Dockers containers automatically, and, again, a simple guide will help site owners to configure the very few required initial parameters (e.g. the IP address of the Robot Framework).

The design of the Testing Tool is highly modular, allowing the easy extension of the base design with necessary functionalities from future use cases. Integration of external tools as well as additional instrumentation can be added using custom exporters and common communication interfaces depending on the needs of each site and use case.

3.3 Testing tools available at 5G EVE site facilities

This section presents the testing tools that are available in the sites and they are used by the site owners for the testing of their site capabilities. Some of them (e.g. iperf) can be replicable to all site since are based on open well know solutions, while others (e.g. video tasting tools) are not necessarily replicable tool since are commercial and specialised solutions. On the contrary, the testing tools described on the above sections, which fulfil the main objective of this deliverable can be replicated to all sites (e.g. deployed as VMs).

3.3.1 Spanish site facility

In the Spanish site we are following the roadmap defined in the 5G EVE project to provide the required testing tools, which will be used by the Spanish verticals in the first place, and all incoming verticals in the future. In order to accomplish this task, we have followed a plan focused on providing specific testing tools for the Spanish use cases in month 10, which concurs with this deliverable D5.1 and project milestone 5 (MS5). The integration of all these tools will occur in a second phase in month 18 when, as declared as MS8, the 5G EVE end-to-end facility will be open to other verticals. Finally, in a third phase after month 18, we plan to incorporate other testing tools to our infrastructure, depending on the necessities of new use cases.

In order to present the testing tools available in the Spanish site, this section will follow this temporal plan, starting with the tools available at month 10, and then explaining the integration plan proposed in a second stage. Because the third phase depends on the upcoming new use cases, these tools will be documented in another deliverable.

3.3.1.1 Existing testing tools

All existing testing tools available in the Spanish site can be classified in three groups: radio spectrum tools, traffic generators and experiment definition and execution.

- ***Radio spectrum tools***. Due to the requirements of the ASTI use case, the Spanish site has deployed an ElectroSense (https://electrosense.org/) node, which can be used to monitor available and used spectrum at one particular site. This project provides open source software and hardware to compose a node. The hardware is based on an ARM computer (Raspberry Pi 3 is the recommended choice) and a Software Defined Radio (SDR) card. The project has an open API to access the information sensed by the node, which can be used to collect spectrum information during the execution of a given experiment.
- ***Traffic generators***. The Spanish site has decided to use the Trafic traffic generator delivered by the Mami project (https://github.com/mami-project/trafic). This generator is based on the well-known iperf3 generator (https://iperf.fr/), extending its main functionalities with other tools like a model to define new traffic profiles, capacity to export logs to CSV/JSON/InfluxDB and an open API to configure an experiment based on Webhooks. As the results of our work in the 5G EVE project, we have two main contributions currently implemented, and a plan to extend it to integrate this VNF with the 5G EVE metrics distribution based on Kafka (see Figure 14):
    - OSM-compliant VNFs implementing the Trafic generators. Our plan in the future is to adapt this VNFs to the 5G EVE descriptors to be fully compliant with the 5G EVE infrastructure.
    - Two Trafic profiles defining both the information generated by ASTI Autonomous Guided Vehicles (AGVs) with sensor information and the commands received by the AGVs in the opposite direction.
    - Our plan is to include a Beats module (Filebeat, more specifically) to integrate this VNF with the 5G EVE metrics distribution, to publish these results in a Kafka bus.

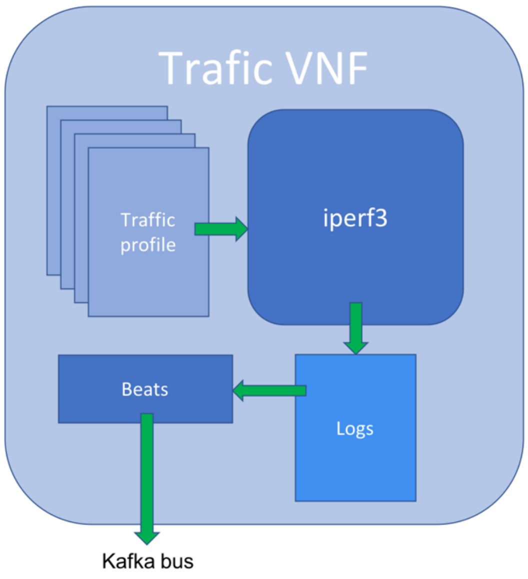


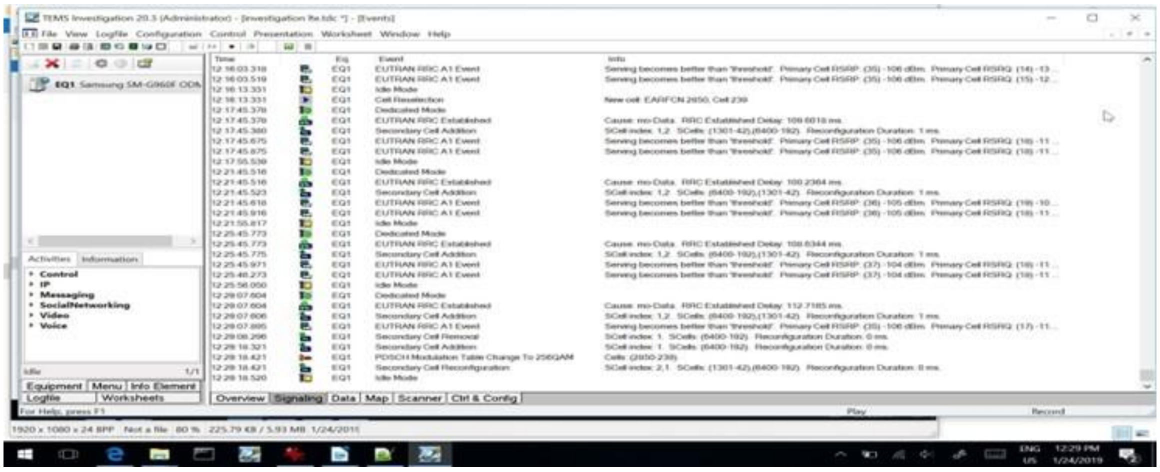
Figure 14: Trafic VNF

- ***Experiment definition and execution (EDE)***. These tools are used to define and run experiments, including the capacity to analyse the result, and are the particularization for 5TONIC of the tools described in section **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**. Figure 11 shows all blocks involved in the testbed used to run preliminary experiments for the ASTI use case. The most important block in such figure is the RobotFW+Jenkins one, which includes a Robot Framework controller (https://robotframework.org/) and Jenkins (https://jenkins.io/), which is used for automation purposes. This block allows the definition of experiments using a human-friendly format. This experiment definition involves the identification of all elements involved in the experiment, the actions that have to be taken and the expected results in each test. For example, in Figure 11 the iperf/ping and the Linux Traffic Control blocks show the main elements involved in the experiment: we plan to extract bandwidth and delays observed at the iperf/ping blocks in several tests, based on the modification of the delay at the Linux Traffic Control block. A python script (also tested in Java language) extracts the metrics at all devices involved in the experiments to publish them using an Apache Kafka bus, which in turn can be processed by a Kafka Exporter block. This information is then stored in a Prometheus Metrics database and visualized in a Grafana web frontend. We have a plan to extract ASTI Key Performance Indicators (KPIs) from their devices, publishing their metrics in the Kafka bus, which could be analysed in turn by the Robot Framework block.

3.3.1.2 Testing tools for Multimedia measurements

Tools mentioned in the previous section are generic tools, applicable to many use cases at 5TONIC. The Multimedia use case, however, demands some other specific tools which are described hereafter:

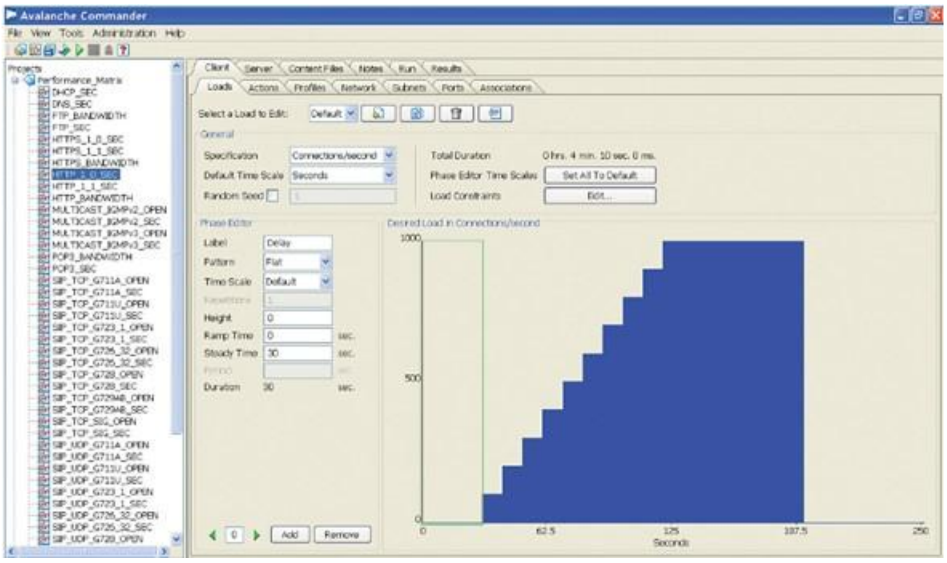
- Instrumental 4G mobile device for radio measurements of the control plane latencies. The selected equipment is the **TEMS Investigation**, currently with an instrumented Samsung S9 terminal. The TEMS Investigation provides low level information and can support some instrumental automatic test that can be programmed. TEMS Investigation is a market-leading end-to-end network testing solution, allows to test every new function and feature in the network. This allows to better understand Customer Experience and to verify, optimize and troubleshoot the mobile network. Through the close cooperation with equipment vendors, chipset manufactures and device vendors the instrumented S9 device is able to execute automated tests. This allows us to quickly provide in-depth subscriber (QoE) and the network (QoS) insights to enable you to make better network investment choices. Whether the purpose is rolling out a new network technology such as LTE or 5G, implementing a new network service like NB-IoT or Video delivery, or optimizing an existing mobile infrastructure, TEMS Investigation gets the job done right the first time. Through close cooperation with chipset manufacturers and scanner vendors, TEMS Investigation have been leading the way to 4G and 5G. Only Infovista, offers you a complete turnkey solution - RF software, geodata and network testing - to cover the entire network planning project. When it is required to enhance the subscriber QoE, the best way to test the network is to assess how subscribers actually experience the network. TEMS Investigation performs on-mobile service and application testing coupled with layer 1-3 data collection to provide you the genuine insights needed to boost your network.



```
TEMS Investigation 20.3 (Administrator) - [Investigation file.tdc *] - [Events]
File  View  Logfile  Configuration  Control  Presentation  Worksheet  Window  Help

EQ1 Samsung SM-G960F ODM

Time          Eq   Event                                    Info
12:16:03.318  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (35) -106 dBm. Primary Cell RSRQ: (14) -13 ...
12:16:03.519  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (35) -106 dBm. Primary Cell RSRQ: (15) -12 ...
12:16:13.331  EQ1  Idle Mode
12:16:13.331  EQ1  Cell Reselection                         New cell: EARFCN 2850, Cell 239
12:17:45.378  EQ1  Dedicated Mode
12:17:45.378  EQ1  EUTRAN RRC Established                   Cause: mo-Data. RRC Established Delay: 108.6818 ms.
12:17:45.380  EQ1  Secondary Cell Addition                  SCell index: 1,2. SCells: (1301-42),(6400-192). Reconfiguration Duration: 1 ms.
12:17:45.675  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (35) -106 dBm. Primary Cell RSRQ: (18) -11 ...
12:17:45.875  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (35) -106 dBm. Primary Cell RSRQ: (18) -11 ...
12:17:55.538  EQ1  Idle Mode
12:21:45.516  EQ1  Dedicated Mode
12:21:45.516  EQ1  EUTRAN RRC Established                   Cause: mo-Data. RRC Established Delay: 100.2364 ms.
12:21:45.523  EQ1  Secondary Cell Addition                  SCell index: 1,2. SCells: (6400-192),(1301-42). Reconfiguration Duration: 1 ms.
12:21:45.618  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (36) -105 dBm. Primary Cell RSRQ: (19) -10 ...
12:21:45.916  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (36) -105 dBm. Primary Cell RSRQ: (18) -11 ...
12:21:55.817  EQ1  Idle Mode
12:25:45.773  EQ1  Dedicated Mode
12:25:45.773  EQ1  EUTRAN RRC Established                   Cause: mo-Data. RRC Established Delay: 108.6344 ms.
12:25:45.775  EQ1  Secondary Cell Addition                  SCell index: 1,2. SCells: (6400-192),(1301-42). Reconfiguration Duration: 1 ms.
12:25:45.971  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (37) -104 dBm. Primary Cell RSRQ: (18) -11 ...
12:25:46.273  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (37) -104 dBm. Primary Cell RSRQ: (18) -11 ...
12:25:56.050  EQ1  Idle Mode
12:29:07.604  EQ1  Dedicated Mode
12:29:07.604  EQ1  EUTRAN RRC Established                   Cause: mo-Data. RRC Established Delay: 112.7185 ms.
12:29:07.606  EQ1  Secondary Cell Addition                  SCell index: 1,2. SCells: (6400-192),(1301-42). Reconfiguration Duration: 1 ms.
12:29:07.895  EQ1  EUTRAN RRC A1 Event                      Serving becomes better than 'threshold'. Primary Cell RSRP: (35) -106 dBm. Primary Cell RSRQ: (17) -11 ...
12:29:08.298  EQ1  Secondary Cell Removal                   SCell index: 1. SCells: (6400-192). Reconfiguration Duration: 0 ms.
12:29:18.321  EQ1  Secondary Cell Addition                  SCell index: 1. SCells: (6400-192). Reconfiguration Duration: 0 ms.
12:29:18.421  EQ1  PDSCH Modulation Table Change To 256QAM  Cells: (2850-239)
12:29:18.421  EQ1  Secondary Cell Reconfiguration           SCell index: 2,1. SCells: (1301-42),(6400-192). Reconfiguration Duration: 0 ms.
12:29:18.520  EQ1  Idle Mode

Overview | Signaling | Data | Map | Scanner | Ctrl & Config
1920 x 1080 x 24 BPP   Not a file   80 %   225.79 KB / 5.93 MB   1/24/2019
```

Figure 15: Instrumental 4G mobile device

- Spirent Avalanche with 80 Gbps video traffic emulation capacity. Spirent’s Avalanche Layer 4-7 testing solution provides capacity, security and performance testing for network infrastructures, cloud and virtual environments, Web application infrastructures and media services that ensures Quality of Service (QoS) and Quality of Experience (QoE) for your customers. Avalanche pinpoints network capacity with the highest performing Layer 4-7 traffic generation capability available from 1Gbps to 100Gbps. Up to 100Gbps of Stateful Traffic Generation—Spirent Avalanche provides the capability to generate stateful traffic from 1Gbps to 100Gbps rates via native interfaces for scale and performance testing for the most demanding of environments. Multiple systems can be used in a single test creating vast amounts of application and security traffic:
    - Flexible Load Profile—Spirent Avalanche’s flexible load profile offers testers the flexibility to specify load variables such as connections per second, concurrent connections, transactions per second, new user sessions per second and concurrent users.
    - Real-Time Statistics—Spirent Avalanche offers real-time statistics across all protocols and reporting tool that makes analysing results faster and easier. Application Server Performance Testing—Spirent Avalanche can validate performance of several types of real servers including: Web Server, Application Server, Mail Server, DHCP Services, FTP Server, DNS Server, Telnet Server, RTSP/RTP QuickTime Streaming Server, Multicast Server, HTTP Adaptive Bitrate Streaming, and more.
    - Media Testing—Spirent Avalanche can perform media testing and service verification from the user’s point of view with realistic voice calling, HTTP Adaptive Bitrate streaming, unicast and multicast streaming video, and simulate Internet data traffic using static IP address or IP address assign by DHCP over PPPoE, VLAN and Stacked VLANs (e.g., Q-in-Q)



Avalanche Commander

File View Tools Administration Help

Projects
- Performance_Matrix
  - DHCP_SEC
  - DNS_SEC
  - FTP_BANDWIDTH
  - FTP_SEC
  - HTTPS_1_0_SEC
  - HTTPS_1_1_SEC
  - HTTPS_BANDWIDTH
  - HTTP_1_0_SEC
  - HTTP_1_1_SEC
  - HTTP_BANDWIDTH
  - MULTICAST_IGMPv2_OPEN
  - MULTICAST_IGMPv2_SEC
  - MULTICAST_IGMPv3_OPEN
  - MULTICAST_IGMPv3_SEC
  - POP3_BANDWIDTH
  - POP3_SEC
  - SIP_TCP_G711A_OPEN
  - SIP_TCP_G711A_SEC
  - SIP_TCP_G711U_OPEN
  - SIP_TCP_G711U_SEC
  - SIP_TCP_G723_1_OPEN
  - SIP_TCP_G723_1_SEC
  - SIP_TCP_G726_32_OPEN
  - SIP_TCP_G726_32_SEC
  - SIP_TCP_G728_OPEN
  - SIP_TCP_G728_SEC
  - SIP_TCP_G729AB_OPEN
  - SIP_TCP_G729AB_SEC
  - SIP_TCP_SIG_OPEN
  - SIP_TCP_SIG_SEC
  - SIP_UDP_G711A_OPEN
  - SIP_UDP_G711A_SEC
  - SIP_UDP_G711U_OPEN
  - SIP_UDP_G711U_SEC
  - SIP_UDP_G723_1_OPEN
  - SIP_UDP_G723_1_SEC
  - SIP_UDP_G726_32_OPEN
  - SIP_UDP_G726_32_SEC
  - SIP_UDP_G728_OPEN

Client | Server | Content Files | Notes | Run | Results

Loads | Actions | Profiles | Network | Subnets | Ports | Associations

Select a Load to Edit: Default

General

Specification: Connections/second
Default Time Scale: Seconds
Random Seed: ☐ 1
Total Duration: 0 hrs. 4 min. 10 sec. 0 ms.
Phase Editor Time Scales: Set All To Default
Load Constraints: Edit...

Phase Editor

Label: Delay
Pattern: Flat
Time Scale: Default
Repetitions: 1
Height: 0
Ramp Time: 0 sec.
Steady Time: 30 sec.
Period: sec.
Duration: 30 sec.

Add | Remove

Figure 16: Spirent Avalanche

- Latency test measurements (source code). Based on the defined User Plane Latency in NGNM ALLIANCE (01/2018) DEFINITION OF THE TESTING FRAMEWORK FOR THE NGMN 5G PRE-COMMERCIAL NETWORKS TRIALS. Version 1.0 (CLEAREST MAPPING TO OUR CASE), the latency can be measured between the UE and the MEC or any other Application Server. The provided code will perform multiple (100) DL/UL standard size (32, 1500B) pings over the UE under test to a local test server, and record Max, Min and Average ping times.

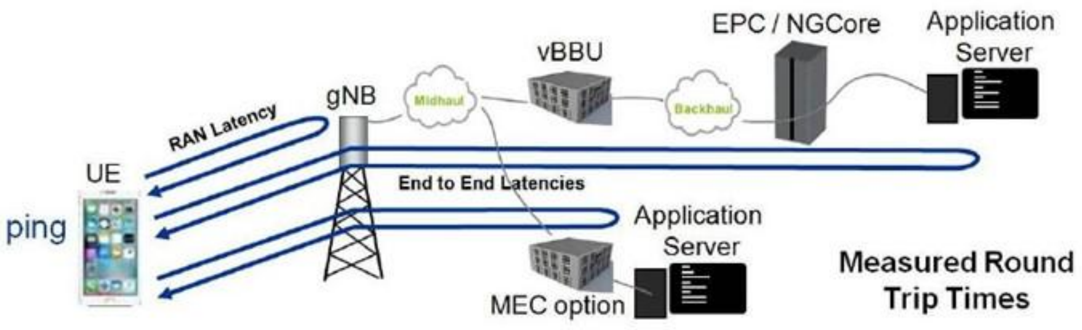


Figure 17: Latency test measurements

- Bandwidth test measurements (source code). Based on the defined User Throughput, Peak Throughput in NGNM ALLIANCE (01/2018) DEFINITION OF THE TESTING FRAMEWORK FOR THE NGMN 5G PRE-COMMERCIAL NETWORKS TRIALS. Version 1.0 (CLEAREST MAPPING TO OUR CASE), the bandwidth can be measured executing download to the UE using UDP or TCP (using iperf) for DL peak test. If possible, measure PDCP layer throughput (PDCP throughput is payload bits divided by the aggregated time) and L1 throughput using measurement tool; session duration should be at least 2 minutes; max./min./avg. value should be reported. There should be only one UE in the cell under test, positioned in the ideal radio conditions. No other UEs should be placed in the surrounding cells. The test can be conducted in indoor or outdoor deployment scenario. Peak rate test should be executed in stationary scenario.

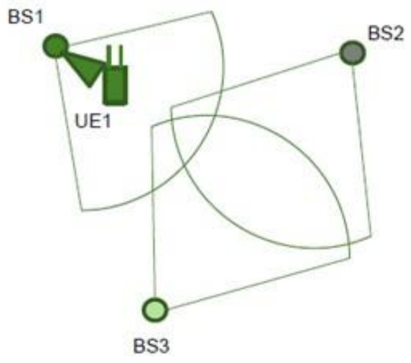
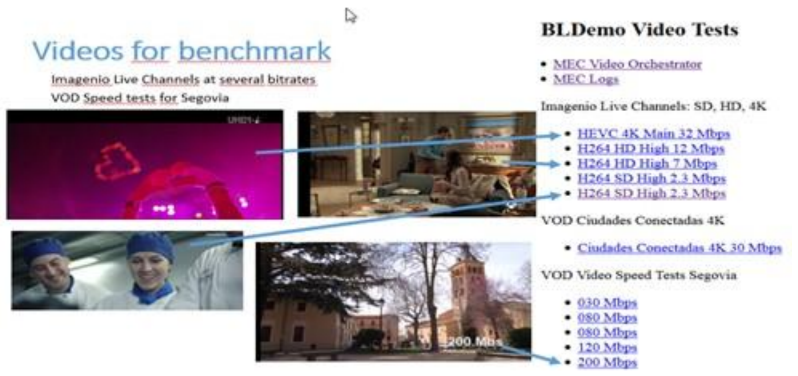


Figure 18: Bandwidth test measurements

- Virtual MEC for video application and delivery. The MEC with encoded video reference applications will include Adaptive Streaming Video samples encoded with several codecs and at several bitrates. This MEC is virtualized implementing the Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification and the VNF Package specifications, defined in ETSI GS NFV-SOL 001 and in ETSI GS NFV-SOL 004. This TOSCA Simple Profile in YAML [3], fulfilling the requirements specified in ETSI GS NFV-IFA 011 [1] and ETSI GS NFV-IFA 014 [2] for a Virtualised Network Function Descriptor (VNFD), a Network Service Descriptor (NSD) and a Physical Network Function Descriptor (PNFD). The present document also specifies requirements on the VNFM and NFVO specific to the handling of NFV descriptors based on the TOSCA. The TOSCA YAML CSAR file is an archive file using the ZIP file format whose structure complies with the TOSCA Simple Profile YAML v1.1 Specification [2]. The CSAR file may have one of the two following structures:
    - CSAR containing a TOSCA-Metadata directory, which includes the TOSCA.meta metadata file providing an entry information for processing a CSAR file as defined in TOSCA v1.0 Specification [i.1].
    - CSAR containing a single yaml (.yml or .yaml) file at the root of the archive. The yaml file is a TOSCA definition template that contains a metadata section with template_name and template_version metadata. This file is the CSAR Entry-Definitions file.


Videos for benchmark

Imagenio Live Channels at several bitrates

VOD Speed tests for Segovia

BLDemo Video Tests

- MEC Video Orchestrator
- MEC Logs

Imagenio Live Channels: SD, HD, 4K

- HEVC 4K Main 32 Mbps
- H264 HD High 12 Mbps
- H264 HD High 7 Mbps
- H264 SD High 2.3 Mbps
- H264 SD High 2.3 Mbps

VOD Ciudades Conectadas 4K

- Ciudades Conectadas 4K 30 Mbps

VOD Video Speed Tests Segovia

- 030 Mbps
- 080 Mbps
- 080 Mbps
- 120 Mbps
- 200 Mbps

Figure 19: CSAR containing a single yaml

- Android application for measurements for video applications. These applications can be used for playing videos on UEs based on Androids, but also to making real-time measurements in android devices. The measurements are recollected in one application running in the MEC and aggregated for further analysis. Data as the cellid, video effective download speed, radio signal level and the GPS position among others is reported in real time to the collection data system that will be used for KPI generation.

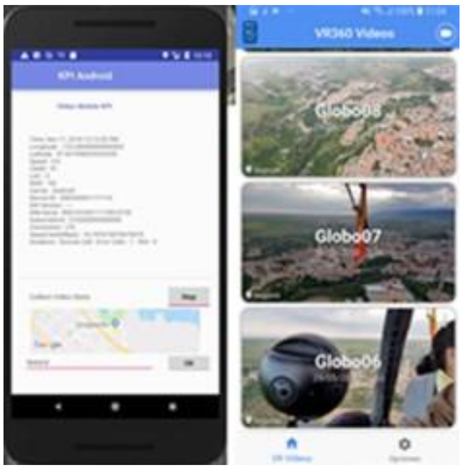


Figure 20: Android application for measurements

- Real time Monitoring tool for video application. This ad-hoc application can be used in real time to display the evolution of any individual UE running on the system and integrated with the MEC collection system. Part of the information is automatically collected from the UE, but other information is collected from the Video Streaming Application Server. Data as the radio signal level, frequency used for the experiment, user data throughput or latency can be monitored in real time while one experiment is taking place, all with real time position monitoring.

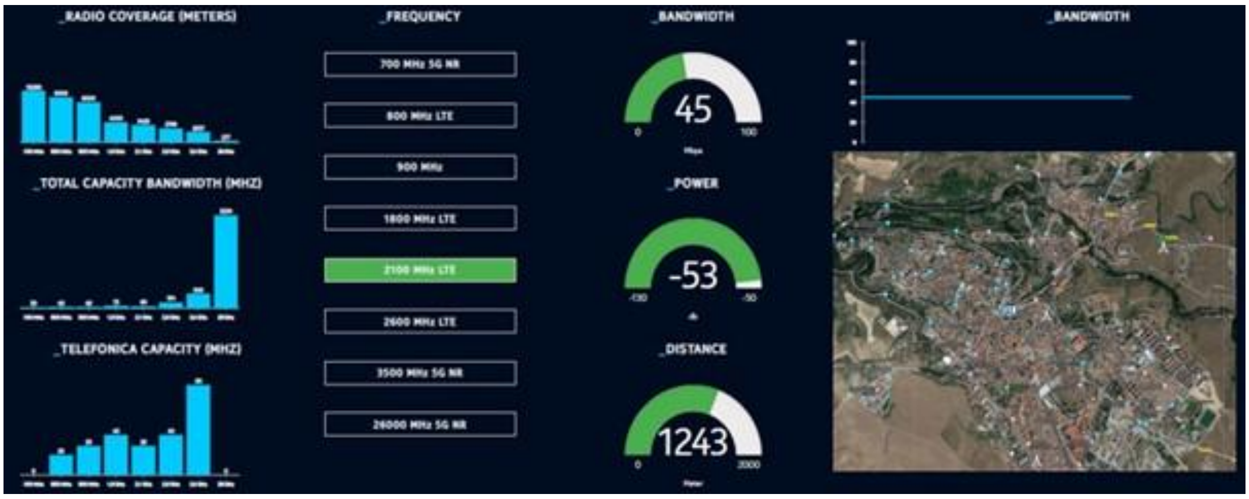


Figure 21: Real time Monitoring tool

3.3.1.3 Integration plan

The Spanish facility operation team has defined a plan to integrate all aforementioned tools, to create a testing environment together with the available elements to deploy all use cases, like 4G/5G routers/antennas, the EPC, the transport network, the OSM MANO stack, the compute nodes, etc. This integrated environment can be seen in Figure 22, where the testing tools and all elements required for testing purposes are shown in light-blue colour, the experiment definition and execution (EDE) blocks and the corresponding configuration network is in green colour, while the metrics distribution framework based on Kafka is in red colour. It is important to highlight that some of these elements can be virtualised as part of the experiment deployment, like the Trafic generators, using the VNF provided by the Spanish site. The Spanish site plans to deliver a high-performance Open Virtual Switch (OVS) as a VNF as part of T4.5, so this element could be virtualised too.

The following is a particularization example showing how the generic tools above will assist in the definition, execution and monitoring of an ASTI experiment. This experiment is focused on extracting ASTI KPIs based on several parameters, like the number of AGVs and the experienced delay in the traffic exchanged between the AGVs at a factory (top-left side of Figure 22) and the master controller at the edge (top-right side of Figure 22).

The first step is to define all tests involved in this experiment, using the Robot Framework and Jenkins format. Tests may include a baseline scenario, where only physical AGVs are involved with no extra delay inserted by the testing environment. A second test may include extra delays in a given range. A third test may include emulated traffic generated by the Traffic devices but with no extra delay. Other tests could be defined by mixing the previous tests.

After the experiment is defined, the second step is to execute such tests, where the EDE will contact all involved elements to configure everything that is necessary. In order to sequentially execute each test, the EDE may have to contact again some elements using again the experiment configuration network. Some elements may need adaptation components, which will translate the EDE API (Robot Framework in our scenario) to the proper interface of the non-compliant component, as in the case of the BBUs in the 5TONIC site.

While the experiments are running, capable elements will publish their correspondent metrics using the Kafka Bus, which will be available to all allowed services, in particular the ELK (Elasticsearch, Logstash and Kibana) stack to monitor and store all information generated during the experiment. Again, those components which are not capable to deliver their metrics with our API, would require an adaptation layer to perform this task.

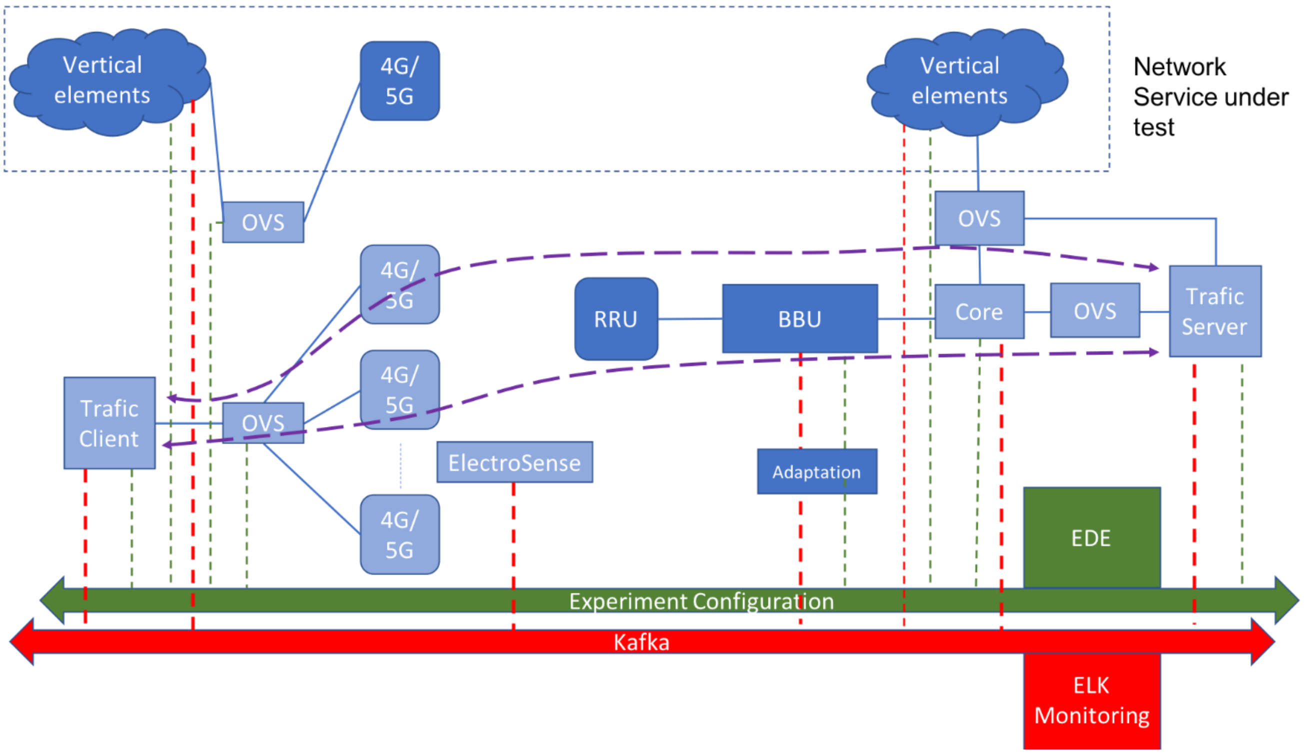


Figure 22. Integrated testing environment in the Spanish site in month 18

3.3.2 Italian site facility

The Italian site is committed to provide the necessary testing tools for 5G EVE verticals, specifically those that are based, or require services, in the area served by the Italian 5G EVE site facilities. The key services will include test consultation, planning, automation, execution and management. While test EPC software will be mainly used in the Core network, and standard microwave/optical/Ethernet links will guarantee backhaul connectivity, on RAN segment, KPI testing will be supported using Spectrum Analysers, emulated handsets and test eNB software. RF shielding environment together with RF attenuators, duplexers, directional couplers and power splitters will allow replicating real conditions in lab environments.

One of the main functionalities is the ability to create different levels of traffic load across multiple cells thus enabling Verticals testing their services to set up a network configuration that replicates both real world conditions and mobility, and also by recreating traffic surges closely similar to those generated by a variety of real smartphones and apps. This is achieved thanks to RF UEs load emulator, as show in Figure 23. Load emulation is crucial because, due to persistent app activities, a typical smartphone ends up connecting to the network countless times every day, resulting in a large amount of signalling traffic in the background. Any network setting and service deployment would have to account for such a burden. Being able to accurately reproduce realistic smartphone traffic patterns leads to a better optimization of network performance and can be used to gauge the end-user experience. The testing equipment in the Italian site allows full load of multiple antenna sectors. On each sector, several emulated UEs concurrently generate load over the radio interface, using SDR units for this task to perform load & stress and functional testing over the radio interface, encompassing complete LTE protocol stacks and their applications. Load and functional testing in LTE/5G NR RAN technology are needed to the task of detecting device-level bugs. In addition, stress and functional testing may help identify the optimal network architecture for a particular service model proposed by the Vertical.

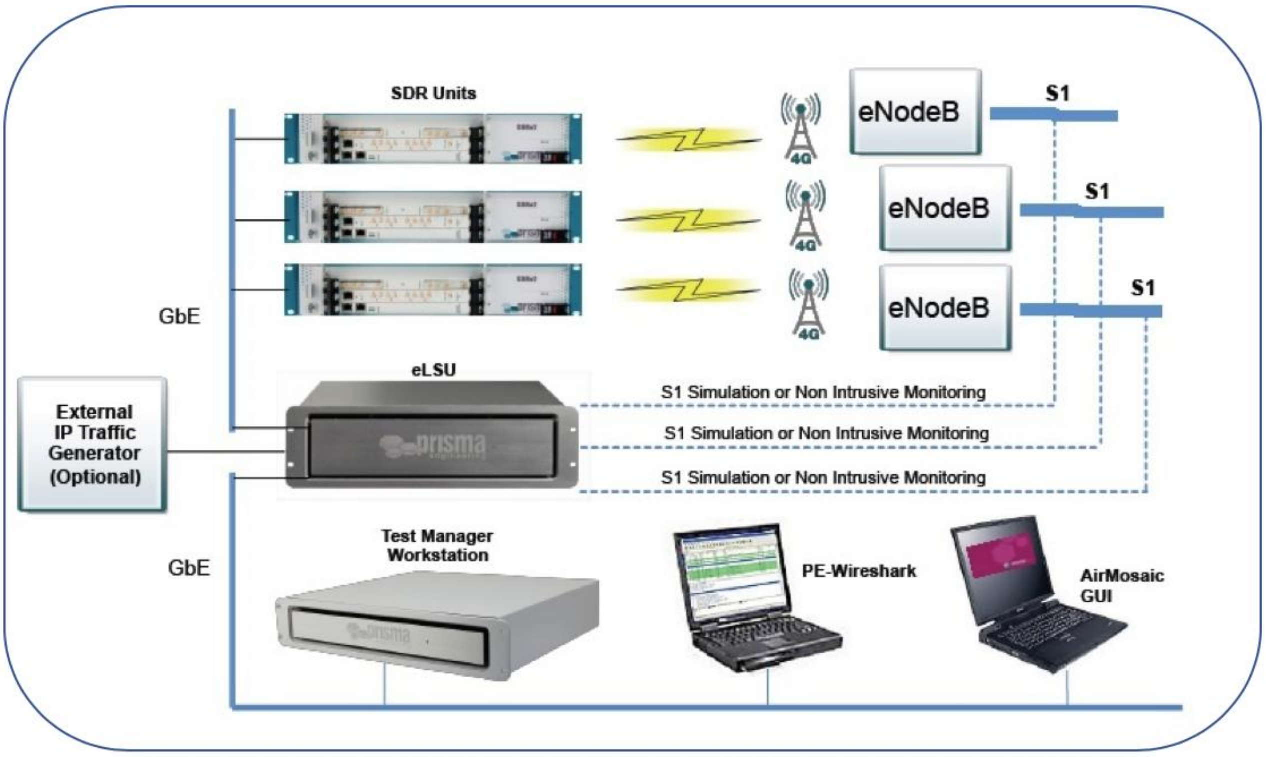

SDR Units
4G
eNodeB
S1
eNodeB
S1
eNodeB
S1
GbE
eLSU
External IP Traffic Generator (Optional)
prisma
S1 Simulation or Non Intrusive Monitoring
S1 Simulation or Non Intrusive Monitoring
S1 Simulation or Non Intrusive Monitoring
GbE
Test Manager Workstation
PE-Wireshark
AirMosaic GUI

Figure 23: Lab testing using PRISMA multi-UE load emulator

The emulated UE can stress both the FDD/TDD radio interface and the Core Network. The available features support Carrier Aggregation (up to 8 carriers) over any FDD/TDD 3GPP band, logical Cells configuration, eMBMS, VoLTE, 2×2/4×4 MIMO (with CA), IoT and more. The configurations are scalable to test most of eNodeB configuration.

The testing configuration in the Italian site is fully manageable by a single control point, shown at the bottom right end of Figure 23, called AirMosaic. Using a GUI that interacts with the Test Manager Workstation, AirMosaic can operate scenarios where the emulated terminals flow seamlessly from one base station to the other, while checking the relevant signalling on the transport interfaces (Uu, S1, X2 for LTE). The AirMosaic GUI also generates graphical reports detailing KPIs and complete performance analysis. Figure 24 shows the architecture of the test scenario generator, encompassing both User plane and Control plane.

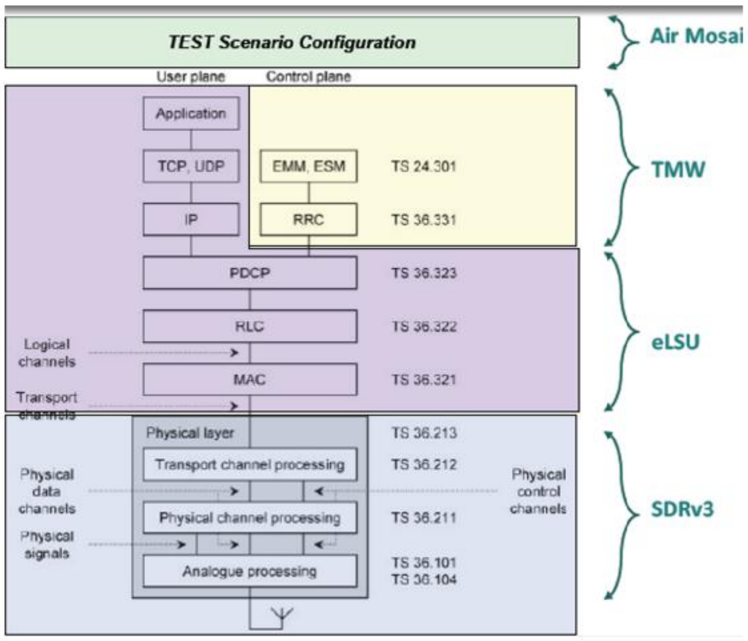


Figure 24: Architecture of the testing suite

The VNF catalogue of the Italian site provides also a number of monitoring and traffic generator tools that can be instantiated in a flexible configuration of virtual instances, as integrated components of the end-to-end service chains of the vertical services. In particular, the following tools are available:

- **iPerf** [1] is a widely used tool for network performance measurement and tuning, providing active measurements of the maximum achievable bandwidth on IP networks. It supports the tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, packet loss, and other parameters. iPerf has client and server functionality and can create data streams to measure the throughput between the two end points in one or both directions.
- **NetEm** [2] is an enhancement of the Linux traffic control facilities that allows to add delay, packet loss, duplication and other characteristics to packets outgoing from a selected network interface. NetEm is built using the existing Quality of Service (QoS) and Differentiated Services (diffserv) facilities in the Linux kernel.
- **Ostinato** [3] is a packet crafter, network traffic generator and analyzer with a friendly GUI. It also offers a powerful Python API for network test automation. Ostinato is able to craft and send packets of several streams at different, configurable traffic rates and it supports several protocols. **Tcpreplay** [4] is a suite of free open source utilities for editing and replaying previously captured network traffic traces.

3.3.3 Greek site facility

3.3.3.1 Existing testing tools

Greek site provides a set of testing tools to be used for experimentation and verification purposes of 5G EVE use cases. The first system validation tool capable of performance evaluation is IXIA IxChariot. This is a client-server based tool that simulates real-world applications, aiming to predict the performance of devices, systems and networks when subjected to real-traffic conditions. The following list summarizes the main benefits of this testing tool:

1. Design for IT teams
   a. Leading assessment tool: access and run tests; multi user out of the box
2. Assess your network with realistic apps
   a. Create realistic applications to validate network performance with enterprise traffic simulation
3. Network SLA verification
   a. End-to-end test of the network to verify SLA commitments
4. Validate hypervisor performance
   a. 100% software endpoints can be deployed to any number of virtual guests to validate VM-to-VM or VM-to-physical networking performance
5. Voice over IP, video traffic, UDP, TCP
   a. Introduces hundreds of real-time voice communication traffic and verify that voice call and video traffic are meeting quality expectations

IxChariot tool comes along with several supporting features listed below:

- 100application scripts simulating enterprise, triple-play, and other Internet traffic
- Real-world application behaviour testing at the transport layer
- Sophisticated traffic patterns with and without quality of service (QoS) tags
- Throughput, jitter, packet loss, end-to-end delay, MOS, and MDI measurements
- Embed custom payloads to test specific data content across the network

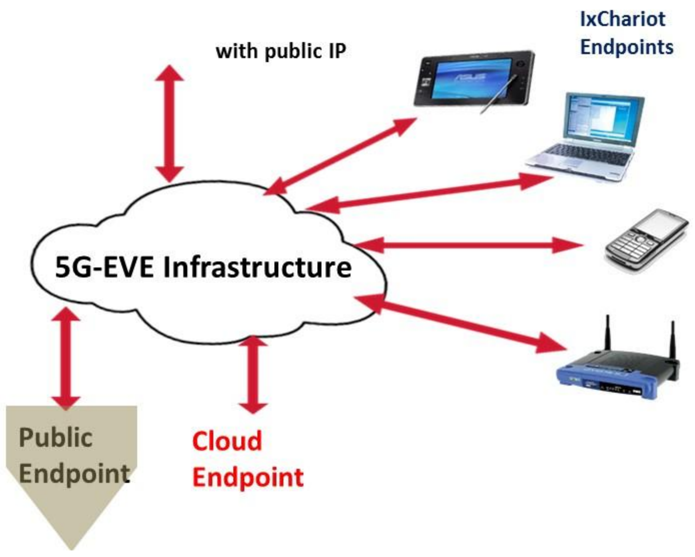


Figure 25: IxChariot topology

IxChariot topology consists of two main parts: (1) A server or console where test performances are configured, and, later, the data collection and reports generation are performed and (2) at least two Performance Endpoints installed in devices included in the test scenario. The endpoints are responsible for initiating traffic, receiving traffic, and sending tdata to the IxChariot console. Figure 25 depicts both parts of this topology. The usage of IxChariot in the 5G EVE infrastructure is shown in Figure 26.

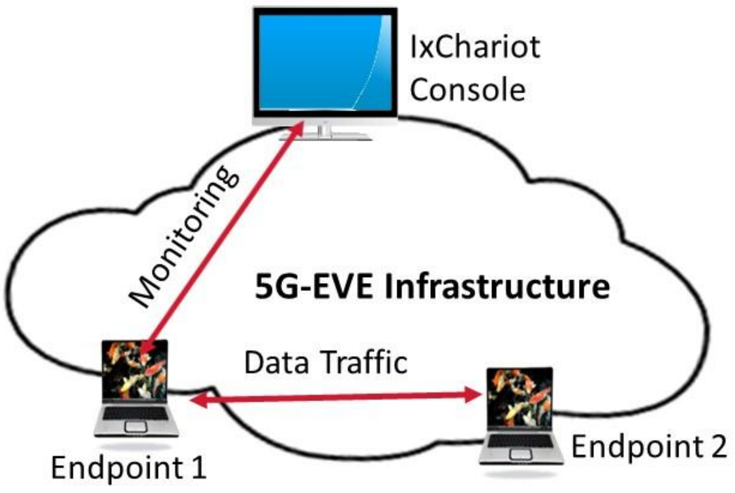
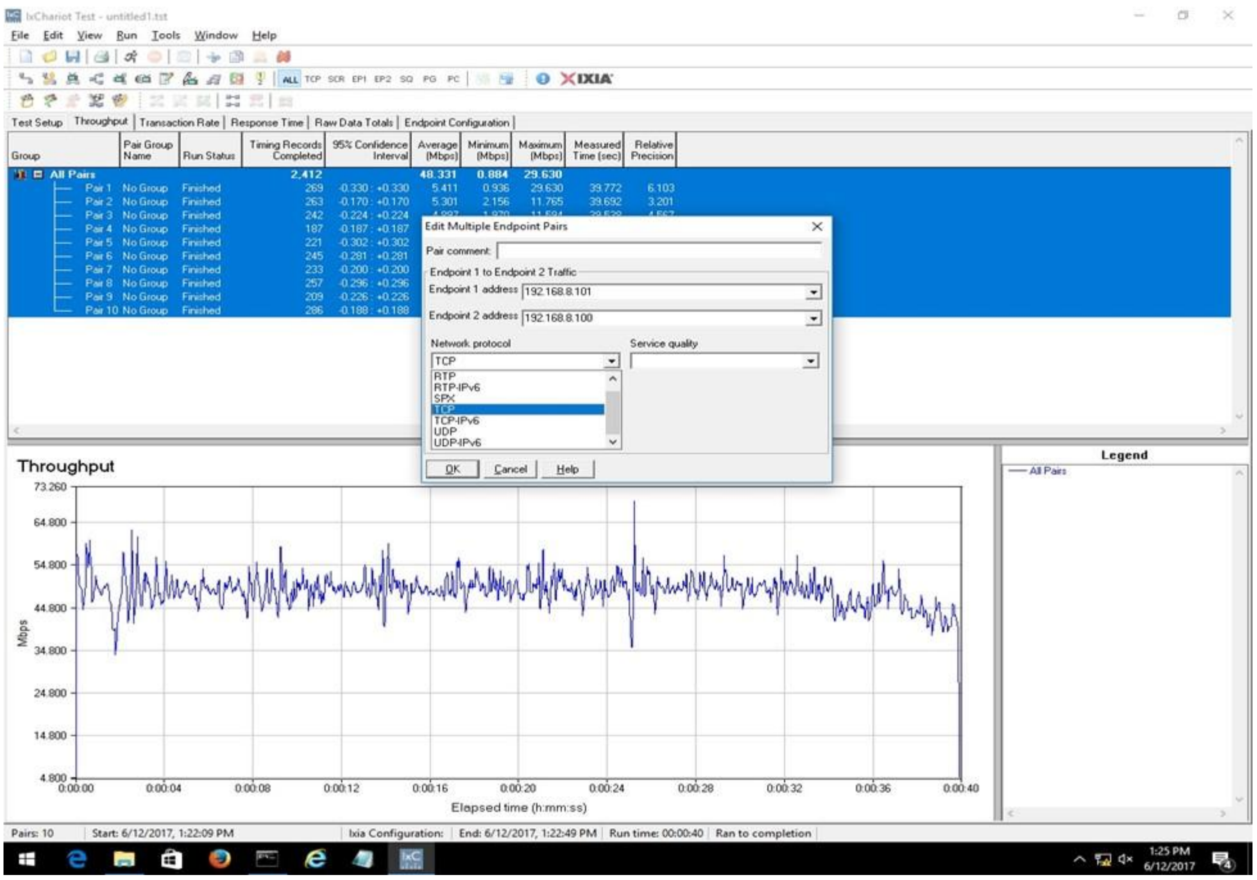


Figure 26: IxChariot in the 5G EVE infrastructure

While the measurements that can be derived are (a) the throughput, (b) jitter, (c) packet-loss, (d) delay and other. Finally, the below screen presents the working environment of IxChariot tool.


IxChariot Test - untitled1.tst
File Edit View Run Tools Window Help
ALL TCP SCR EP1 EP2 SQ PG PC
IXIA
Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration
Throughput
Mbps
Elapsed time (h:mm:ss)
Legend
All Pairs

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			2,412		48.331	0.884	29.630		
Pair 1	No Group	Finished	269	-0.330 : +0.330	5.411	0.936	29.630	39.772	6.103
Pair 2	No Group	Finished	263	-0.170 : +0.170	5.301	2.156	11.765	39.692	3.201
Pair 3	No Group	Finished	242	-0.224 : +0.224					
Pair 4	No Group	Finished	187	-0.187 : +0.187					
Pair 5	No Group	Finished	221	-0.302 : +0.302					
Pair 6	No Group	Finished	245	-0.281 : +0.281					
Pair 7	No Group	Finished	233	-0.200 : +0.200					
Pair 8	No Group	Finished	257	-0.296 : +0.296					
Pair 9	No Group	Finished	209	-0.226 : +0.226					
Pair 10	No Group	Finished	286	-0.188 : +0.188					

Edit Multiple Endpoint Pairs

Pair comment:

Endpoint 1 to Endpoint 2 Traffic

Endpoint 1 address: 192.168.8.101

Endpoint 2 address: 192.168.8.100

Network protocol: TCP

Service quality:

RTP
RTP-IPv6
SPX
TCP
TCP-IPv6
UDP
UDP-IPv6

OK | Cancel | Help

Pairs: 10 | Start: 6/12/2017, 1:22:09 PM | Ixia Configuration: | End: 6/12/2017, 1:22:49 PM | Run time: 00:00:40 | Ran to completion

Figure 27: IxChariot tool

Along with the testing tool, Greek site provides the framework for automating the experimental and verification procedures. Some of the advantages of using Robot framework for this purpose are:

1. Tabular syntax for creating test cases in a uniform way
2. Easy-to-read result reports and logs in HTML format
3. Supports Selenium for web testing, Java GUI testing, running processes, Telnet, SSH, and other protocols
4. Provides tagging to categorize and select test cases to be executed
5. Easy integration with Jenkins for Continuous Integration tasks

The architecture of Robot automation framework is shown below.

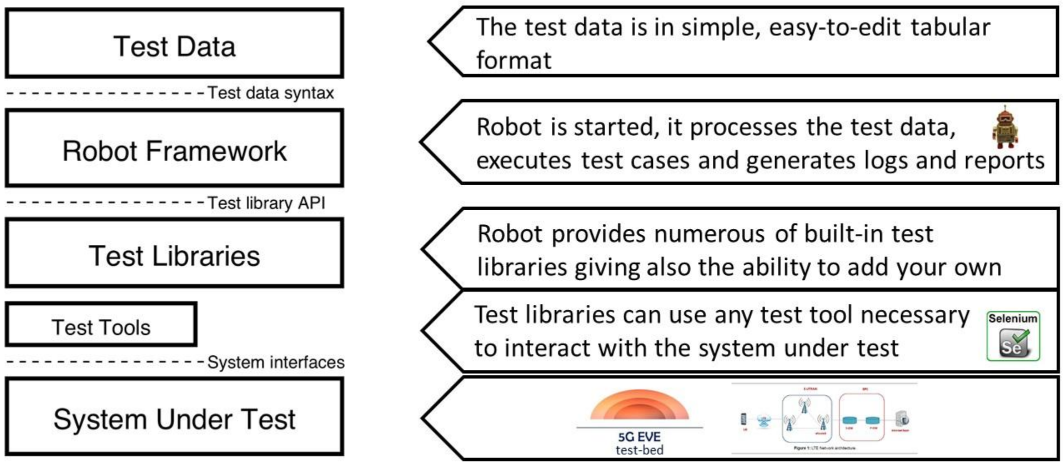

Test Data
Test data syntax
Robot Framework
Test library API
Test Libraries
Test Tools
System interfaces
System Under Test
The test data is in simple, easy-to-edit tabular format
Robot is started, it processes the test data, executes test cases and generates logs and reports
Robot provides numerous of built-in test libraries giving also the ability to add your own
Test libraries can use any test tool necessary to interact with the system under test
Selenium
5G EVE test-bed

Figure 28: Robot framework

Jenkins tool is also available in the Greek site to perform continuous integration and build the automation procedure. Jenkins is integrated with Robot framework initiating test suites on systems-under-test summarizing also the results of each execution in a graphical manner.

In addition, some of the open source testing tools that will be used for performance evaluation are :

- **Robot Framework**: an open source automation framework for acceptance testing . It will be used as the controller of the experiment definition and execution framework.
- **Jenkins**: an automation server that will handle the execution of experiments and the collection of the results
- **Iperf**: a widely used tool for network performance measurement and tuning. It is significant as a cross-platform tool that can produce standardized performance measurements for any network. Iperf has client and server functionality, and can create data streams to measure the throughput between the two ends in one or both directions. Typical Iperf output contains a time-stamped report of the amount of data transferred and the throughput measured.
- **Bmon**: a network monitoring and debugging tool for Unix-like systems, which captures networking related statistics. It is a reliable and effective real-time bandwidth monitor and rate estimator.
- **NetEM**: an enhancement of the Linux traffic control (TC) facilities that allow to add delay, packet loss, duplication and more other characteristics to packets outgoing from a selected network interface. NetEm is built using the existing Quality Of Service (QOS) and Differentiated Services (diffserv) facilities in the Linux kernel.

This set of tools will be enhanced based on specific needs and capabilities of the various use cases.

The KPIs and tools that will be used to assess the performance of the end-to-end system are listed in Table 3.

Table 3: List of testing tools & KPIs available at the Greek 5G EVE site facility

KPI	Definition	Tools
E2E DL latency	End to end mobile network downlink latency	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E UL latency	End to end mobile network uplink latency	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E DL latency variation	End to end mobile network downlink latency variation	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E UL latency variation	End to end mobile network uplink latency variation	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
Uplink buffering delay	Buffering delay introduced by the network on camera streams sent by AGV to main control in uplink direction	Timing difference between packets coming through the mobile network and a direct cable connection to the camera. Wireshark and the dedicated SW developed for the latency analysis will be used
Number of collisions vs. number of obstacles	Ratio between the number of collisions of the AGV with respect to the number of encountered obstacles	Heuristic metric based on observations (use case specific)
Number optimal cruises vs. Total cruises	Ratio between the number of AGV travels following the optimal path with respect to the number of executed tasks	Heuristic metric based on observations (use case specific)

3.3.3.2 Integration plan

The initial efforts with regards to the Testing Framework are focusing around the ASTI use case, taken in WP5 as first example also as it has happened in other WPs. Since this use case is not available on the Greek site another use case has been selected to be used, in this case the Utilities use case. The following is an example about the definition, execution and monitoring of a Utilities experiment. This experiment is focused on extracting utilities service (power consumers and producers) KPIs based on several parameters, like the time need for the control server to decide a new route for the power, the time it takes for the consumer node to be restored to the required voltage and the down time of the consumer node.

The first step is to define all tests involved in this experiment, using the Robot Framework and Jenkins format. Tests may include a baseline scenario, where only the control server and the nodes are involved with no extra delay inserted by the testing environment. A second test may include extra delays in different parts of the service workflow. A third test may include emulated traffic generated by emulated nodes on the network but with no extra delay. Other tests could be defined by mixing the previous tests.

After the experiment is defined, in order to execute the experiment some final configurations will need to be sent to the various element of the experiment. The EDE is responsible for the different stages of the experiment. During the experiments, elements of interest will send metrics over the Kafka Bus, which will be available to the Monitoring, Visualization and Validation tools at first. In the next phase the Kafka Bus will become available to other available services like the common collector.

3.3.4 French site facility

3.3.4.1 TaaS introduction

The French site facility is composed of a cluster of 4 nodes located in different cities whose technology rests on two pillars:

- A pre-commercial 5G platform developed by Nokia Mobile Networks Business Unit,

- a fully open-source solution based on Open Air Interface.One of the 5G EVE ambitions is to deliver Testing-As-A-Service (TaaS) to provide a unified functional and operational infrastructure for vertical industry experimentation. The French site facility will comply with the TaaS requirements. TaaS is inspired by CloudHealth framework[1] developed in the context of NGPaaS project as a solution to configure, deploy and operate the monitoring infrastructure. TaaS provides data probes, data collection and data analysis-as-a-service that collectively result into a built-to-order monitoring environment.

TaaS process is structured in three main stages:

- In the **Configure stage**, the cloud operator selects the monitoring goals (quality attributes that must be computed and observed during operation) and the relevant cloud services. For example, the cloud operator may decide to monitor the reliability and the efficiency of a subset of the services available in the target platform. The choice is automatically mapped by the TaaS Monitoring Model into a set of metrics that must be collected from selected services.
- In **the Deploy stage**, the list of metrics is mapped to a set of probes to be deployed on the target system based on the knowledge of the platform architecture and a probes catalogue. For example, if services to be monitored are executed on virtual machines and the time required to recover from failures must be measured, a probe that performs heartbeat pings (such as pings via ICMP or TCP) can be used. The identified probes can then be deployed and attached to the monitored services. Note that for PNF, the probe is “virtual” as will be detailed below.
- In the **Operate stage**, a dashboard is automatically reconfigured according to the goals to watch both system and services health. This dashboard provides different visualizations at various level of granularity from global system down to the leaves where we find components Throughput, Latency and Response Time.

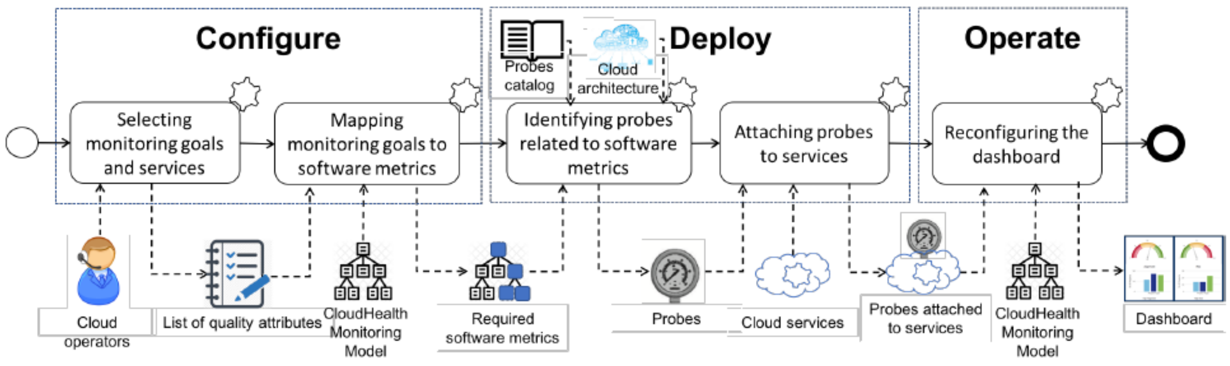

Configure
Deploy
Operate
Probes catalog
Cloud architecture
Selecting monitoring goals and services
Mapping monitoring goals to software metrics
Identifying probes related to software metrics
Attaching probes to services
Reconfiguring the dashboard
Cloud operators
List of quality attributes
CloudHealth Monitoring Model
Required software metrics
Probes
Cloud services
Probes attached to services
CloudHealth Monitoring Model
Dashboard

Figure 29: monitoring process

3.3.4.2 TaaS link to EVE

The dashboard is the starting point for:

- *problem diagnosis*: if a monitoring goal takes an unhealthy state, the related KPIs, which represent software metrics measured on specific resources can be shown for further analysis, possibly automatically.

[1] CloudHealth: A Model-Driven Approach toWatch the Health of Cloud Services

- *Relays to 5G EVE Portal Backend* through IWL NFVO according to Figure 30.

The Monitoring Model is inspired by the ISO/IEC 25010:2011 and ISO/IEC TS 25011:2017 standards that provide de facto specifications of quality models to evaluate the quality of general IT services and are easily adapted to cloud services:

- Monitoring goals are high level monitoring objectives that can be selected by cloud operators
- Monitoring sub goals are high level monitoring objectives that represent the decomposition of monitoring goals.
- Cloud properties, such as memory and CPU consumption, are measurable characteristics of a cloud system, that is, it is possible to measure them using probes attached to services.

The model can be modified by cloud operators to adapt to their needs. In other words, an operator can modify or add new monitoring goals and sub-goals and update the mapping of sub-goals to other cloud properties.

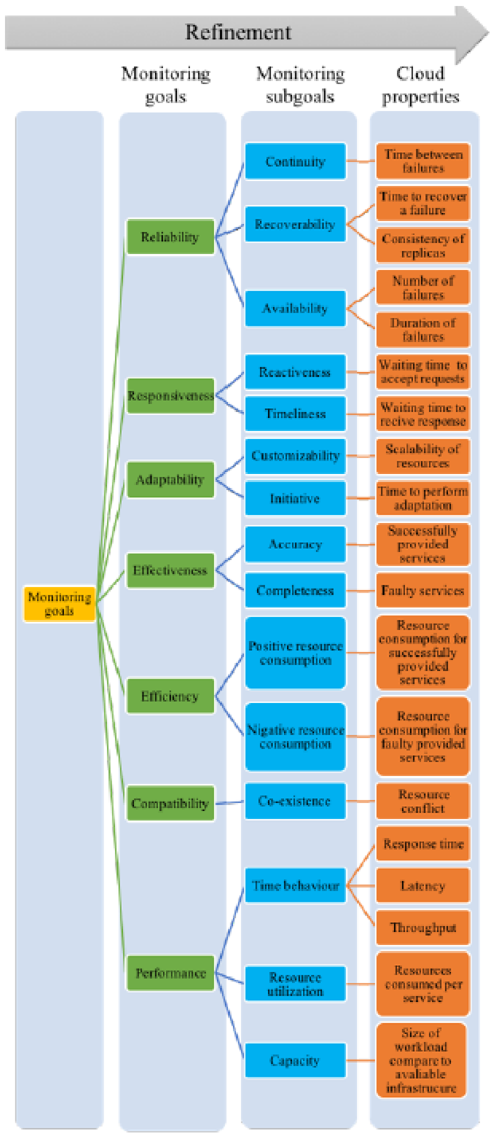


Figure 30: monitoring model

Typical example of use-cases:

1. Use Case 1: Monitor Services at Business Level.
    a. Actor: a manager.
    b. Wanted Feature: he/she wants to monitor cloud services offered by her company at high level of abstraction.

   c. How: TaaS allows her to select monitoring goals at high level of abstractions, automatically generate a set of probes attached to the target services to measure KPIs related to the selected monitoring goals, generate a customized dashboard that allows the actor to monitor KPIs related to her goals and related subgoals at different level of abstractions.

2. Use Case 2: Specified Monitoring for Multi-actors.
   a. Actor: a manager and a technician.
   b. Wanted Feature: they want to monitor the same service at the same time while they have different interests.
   c. How: TaaS offers a customizable model to select monitoring goals. At the technical level, it attaches the set of probes needed to collect the KPIs of interest from the selected services, regardless of the number and type of actors involved. Then, it will provide a customizable dashboard for each actor following the selected goals.
3. Use Case 3: Add New Monitoring Goals.
   a. Actor: a manager or a technician
   b. Wanted Feature: they want the flexibility to redefine the hierarchical decomposition of goals into finer grained characteristics.
   c. How: TaaS is designed to be extendable, which means that the actors can add new monitoring goals. They only need to extend model in terms of monitoring goals, subgoals (if exists) and cloud properties, identify software metrics and their related probes that will be used to collect KPIs about cloud proprieties. The rest of the work will be automatically performed.
4. Use Case 4: Dynamically Switching Between Monitoring Goals.
   a. Role: a manager or a technician.
   b. Wanted Feature: they want monitoring flexibility that allows them to dynamically switch between monitoring goals over time based on current interests.
   c. How: TaaS provides abilities to its users to switch between monitoring goals once needed. Changing the monitoring goals implies automatically deploying a new set of probes and reconfiguring the dashboard.

3.3.4.3 TaaS extension

This model will be extended in 3 directions:

- Pre-commercial equipment
- Open-source prototype
- Hybrid system

3.3.4.3.1. TaaS (pre-)commercial extension

(Pre-)commercial products provide different types of performance monitoring options. First their measured KPIs depend on the nature of the product (segment). Example of mobile KPIs are:

- Sector/cell bitrates: max, average...
- Accessibility
- Drops (UE_drop_rate)
- Mobility
- Quality:
  - RSSI, RSRP, RSRQ, CINR
  - coding: WB CQI QAM16/QAM/64/QPSK…
  - CQI, trafic

Example of transport KPIs are:

- Ethernet links (backhaul, fronthaul, high layer split, eCPRI…) can be tested with a split test analyser.
- Ethernet (including eCPRI) can be tested up to 25Geth.
- Ethernet links can be stressed up to 10Geth.
- CPRI can be analysed up to 10 Gbit/s.

Equipments, whether final product or in-line tester, can usually be accessed via REST APIs. For instance, gNB REST API as presented in Figure 32.

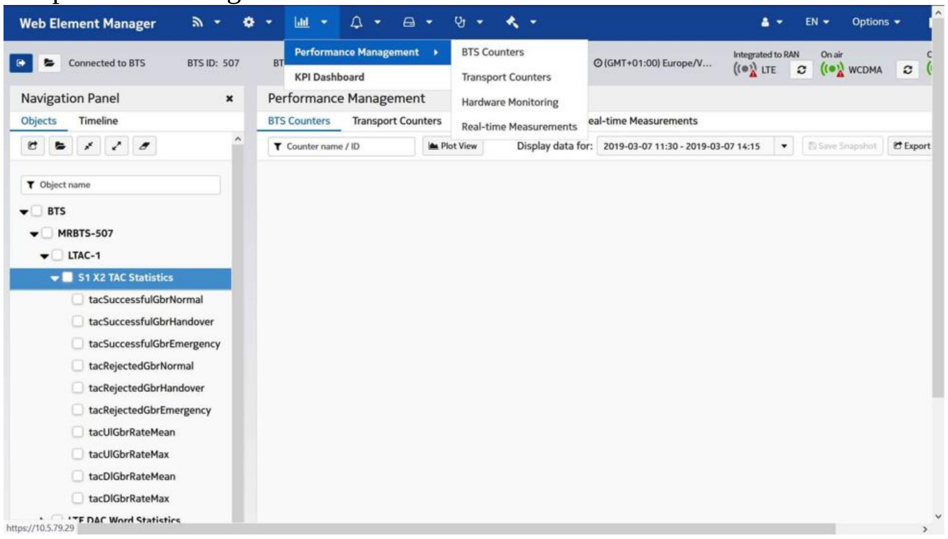


Figure 31: pre-commercial monitoring

Whatever the option, to make these measurements available in TaaS, we need to follow the TaaS extension process as defined in section 3.3.4.2:

- Extend TaaS monitoring model by adding the new goals, sub goals and properties
- Add new corresponding probes: in the case of a REST monitoring, a virtual probe that polls the REST API is created.

Beyond the REST based monitoring, other tools will be added in the process such as:

- gNodeB Traffic Supervision: L1GUI
- Packet Monitoring: Wireshark
- Low level traces analyzer: Sherpa
- Mobile monitoring: QxDM and xCAL

3.3.4.3.1. Open-source extension

The OAI UE and eNodeB software grants full access to all the radio transmission layers allowing to collect and analyse all the data required for vertical experimentation. Wireshark probes will provide the capacity to collect network traffic at the relevant points of the 5G infrastructure. This will enable the KPI collection close to the following table.

Table 4: List of testing tools & KPIs available at the French site facility

KPI	Definition	Tools
E2E DL latency	End to end mobile network downlink latency	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E UL latency	End to end mobile network uplink latency	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E DL latency variation	End to end mobile network downlink latency variation	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed
E2E UL latency variation	End to end mobile network uplink latency variation	Wireshark connected on both ends simultaneously and dedicated analysis SW to be developed

The same process as for the pre-commercial approach will be followed:

- Extend TaaS monitoring model by adding the new goals, sub goals and properties
- Add new corresponding probes: a wireshark-based probe will be used.

3.3.4.3.1. Hybrid extension

Hybrid extensions are new goals, sub goals and properties defined by composing individual (pre-commercial and open-source) together to create compound ones. Again, same approach will be used:

- Extend TaaS monitoring model by adding the new goals, sub goals and properties at the system level
- Add new corresponding probes (if needed)

The dashboard will be used to represent the compounded monitoring.

3.4 Initial testing and validation results

3.4.1 Application in the ASTI use case

As already explained in 3.3.1.1, one important point for the 5G EVE project is to provide testing tools to extend the capabilities of the end-to-end trial platform. For the ASTI use case, one of the main objectives is to run different experiments with emulated AGVs to stress the network to detect any issues before a real scenario with tens of AGVs is deployed in a factory. In order to define an AGV traffic profile, we have defined a test where one AGV is moving around the Spanish trial site, using a 4G mobile network to send information from the AGV to the master PLC, which decodes the information captured by all sensors deployed in the vehicle, processing all this information to generate the proper commands, which are transmitted back to the AGV using the 4G network too (see 2.1.2.1 for the complete definition of the testbed). All this traffic is captured at the master PLC, and shown in Figure 32. After analysing both traffic, the traffic profiles have been described to be executed by the Traffic application.

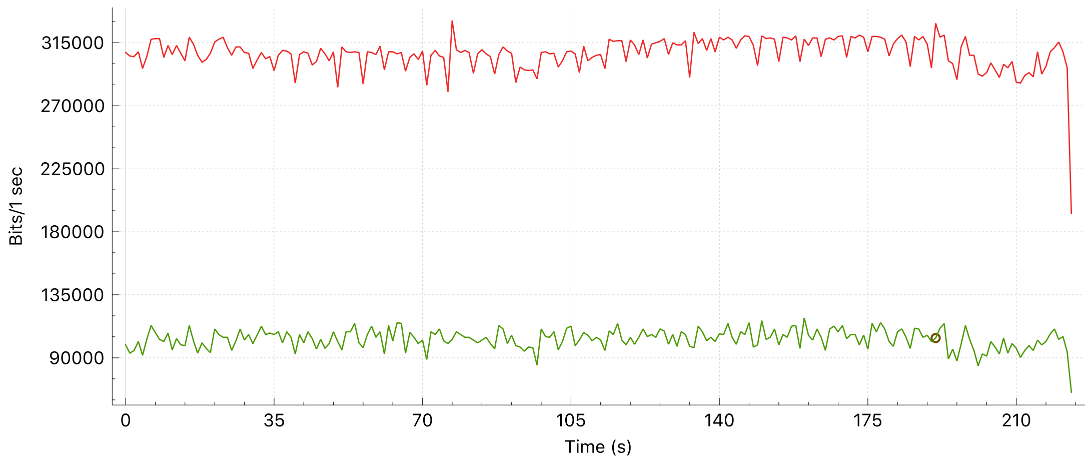


Figure 32: Traffic sent from the AGV (red) and from the master PLC (green)

3.4.2 Application on the Utilities use case

The original proposal is focused around the ASTI use case mainly. Since this use case is not available on the Greek site another use case is selected to be used as an example, in this case the Utilities use case. For the Utilities use case, the main focus of the experiments is to stress the network covering the power grid to detect any issues before implementing the vertical service on a live power grid.

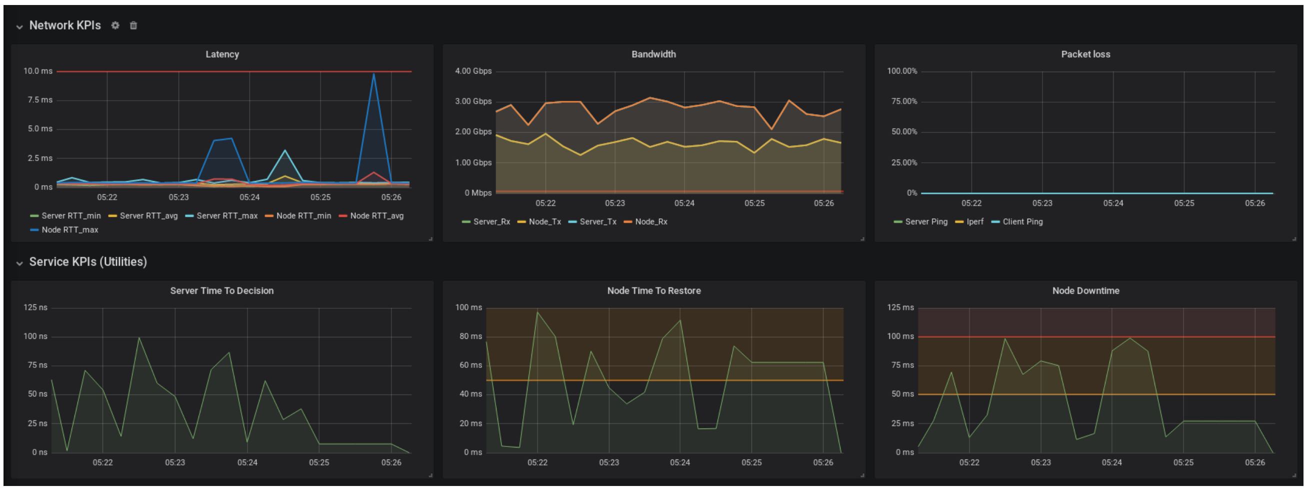


Figure 33: Initial testing on the Utilities use case

In order to define a lifelike scenario, we have defined an example test where a mini power grid is managing the electric circuits of a specific area and is being monitored by a remote service. Additional traffic is inserted in order to stress the link between the remote service and the nodes. Network traffic is captured on the remote service host and on the node. The remote service receives continuous updates on the status of the various nodes of the grid, using a mobile network in order to regulate their operation for optimal performance and also to take appropriate actions in case of critical events. When a critical event, such as a power loss occurs, the remote service intervenes to reroute power to the affected nodes so that the node operation remains unaffected.

3.4.3 Application on the Smart City use case

Another use case supported on the Greek site that is used as an example is the Smart City use case.

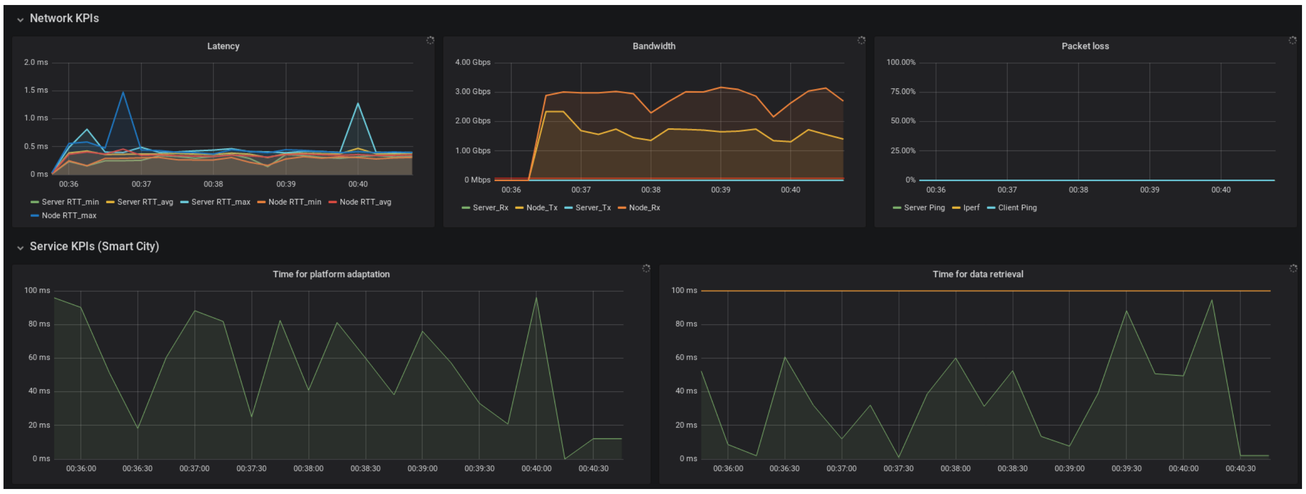


Figure 34: Initial testing on the Smart City use case

For the Smart City use case, the experiment is focused on stressing the network providing the vertical service. In order to test the service on a real scenario, we have defined an example test where the platform is monitoring a multitude of sensors and actuators, using a mobile network, in order to detect as well as predict any events requiring action and handle them accordingly. Additional traffic is inserted in order to stress the link between the remote service and the nodes. Network traffic is captured on the platform host and the node connected to the sensors and actuators. The platform is receiving continuous measurements from the remote nodes and uses predictive mechanisms to create forecasts for future events. In case of an event or a prediction of an event the platform adapts the operation of the various actuators as a response.

3.4.4 Experiment visualization GUI

This chapter provides a description of the first prototype of the 5G EVE Visualization GUI, focusing on the section where the Vertical user can visualize the results of the executed experiments, as elaborated from the Experiment Automation Framework. In this deployment, the EAF is based on Jenkins [5] and Robot Framework [6] tools, while the Visualization GUI is implemented through Robot Metrics [7].

In this initial prototype, the Visualization GUI shows the results of the experiments with different levels of detail and aggregation criteria. In particular, the original results elaborated by Robot Framework during the execution of the various test cases are collected and stored in elementary reports at the EAF. These reports are simple statements that document if the suite, as a whole, is passed or not; moreover, they also identify which test cases are passed within the test suite. Such reports are further processed, still using Robot Framework, to produce an aggregated report that provides the percentage of successful experiment executions, based on three different classification criteria: test suites, test cases (within a single or more test suites) and KPIs (defined as keywords). The Visualization GUI reporting these aggregated results is shown in Figure 35, where for each of the classification criteria, the dashboard provides the statistics for the total, passed and failed number of experiments run.

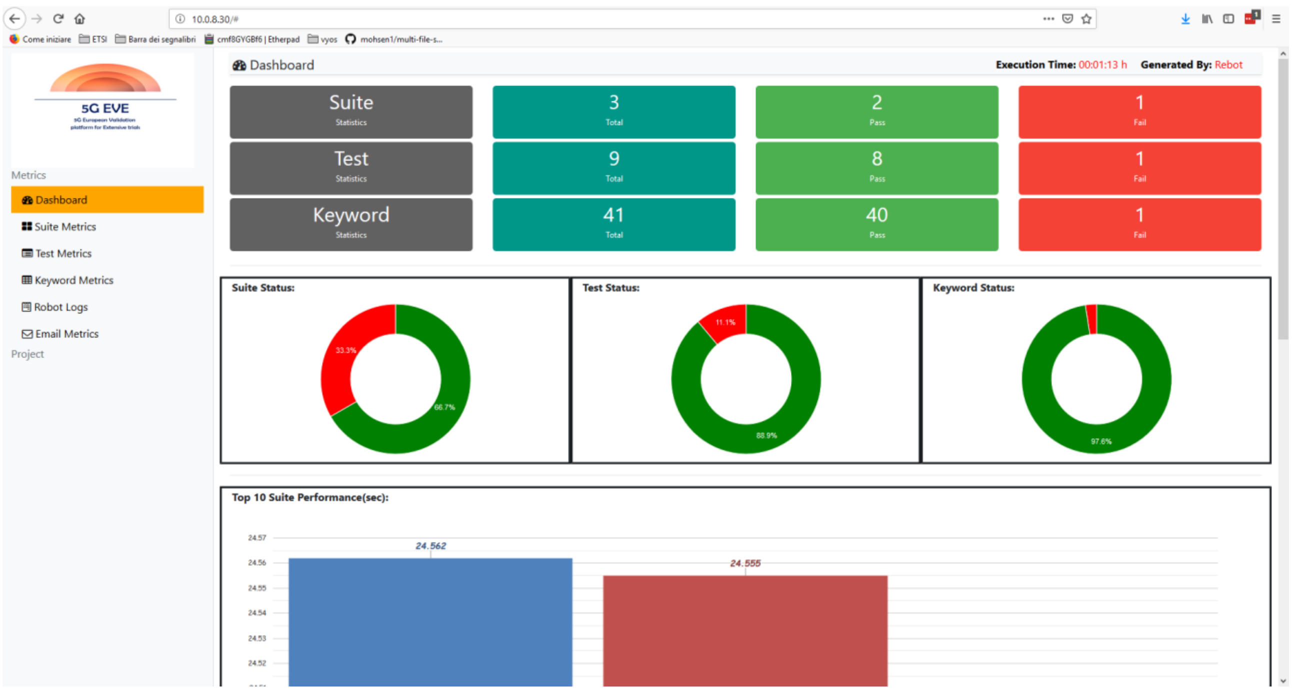
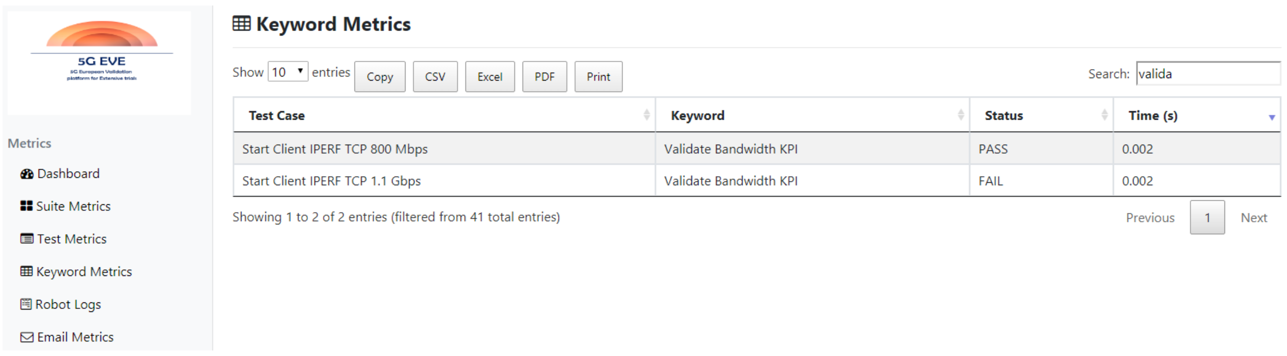


Figure 35: 5G EVE Visualization GUI – Main page

The dashboard provides also details specific of the experiment results for each classification criteria, showing different levels of information. These data can be selected from the menu on the left side of the dashboard. For example, in case of test suites, the dashboard presents the list of the executed experiments and details related to their status and the number of their test cases. In the test cases view, the dashboard offers a more detailed report with the status and result of single test cases. This allows the Vertical user to analyse which particular test case failed, thus better understanding the particular experiment conditions that led to specific kinds of failures. The last view of the dashboard is dedicated to the keywords (see Figure 36), where the Vertical user can visualize all the information related to the KPIs, analysing which ones have failed and in which test cases. The Vertical user can thus identify the conditions that generate the failures. In the example of the figure, a sample test with the Iperf tool shows a successful result with 800 Mbps of traffic and a failure with 1.1 Gbps of traffic.

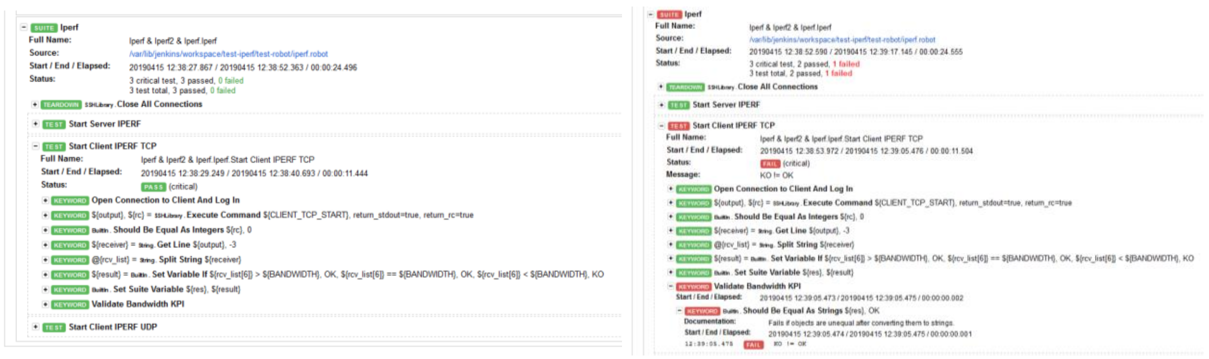


Test Case	Keyword	Status	Time (s)
Start Client IPERF TCP 800 Mbps	Validate Bandwidth KPI	PASS	0.002
Start Client IPERF TCP 1.1 Gbps	Validate Bandwidth KPI	FAIL	0.002

Showing 1 to 2 of 2 entries (filtered from 41 total entries)

Figure 36: 5G EVE Visualization GUI – KPI view

For a deeper analysis, the current version of the dashboard provides also the Robot Framework logs, as originally generated by the tool itself (see Figure 37). This feature is available from the Robot Logs entry in the left side menu.



```
− SUITE Iperf
  Full Name:            Iperf & Iperf2 & Iperf.Iperf
  Source:               /var/lib/jenkins/workspace/test-iperf/test-robot/iperf.robot
  Start / End / Elapsed: 20190415 12:38:27.867 / 20190415 12:38:52.363 / 00:00:24.496
  Status:               3 critical test, 3 passed, 0 failed
                        3 test total, 3 passed, 0 failed
  + TEARDOWN SSHLibrary.Close All Connections
  + TEST Start Server IPERF
  − TEST Start Client IPERF TCP
    Full Name:            Iperf & Iperf2 & Iperf.Iperf.Start Client IPERF TCP
    Start / End / Elapsed: 20190415 12:38:29.249 / 20190415 12:38:40.693 / 00:00:11.444
    Status:               PASS (critical)
    + KEYWORD Open Connection to Client And Log In
    + KEYWORD ${output}, ${rc} = SSHLibrary.Execute Command ${CLIENT_TCP_START}, return_stdout=true, return_rc=true
    + KEYWORD BuiltIn.Should Be Equal As Integers ${rc}, 0
    + KEYWORD ${receiver} = String.Get Line ${output}, -3
    + KEYWORD @{rcv_list} = String.Split String ${receiver}
    + KEYWORD ${result} = BuiltIn.Set Variable If ${rcv_list[6]} > ${BANDWIDTH}, OK, ${rcv_list[6]} == ${BANDWIDTH}, OK, ${rcv_list[6]} < ${BANDWIDTH}, KO
    + KEYWORD BuiltIn.Set Suite Variable ${res}, ${result}
    + KEYWORD Validate Bandwidth KPI
  + TEST Start Client IPERF UDP
```

```
− SUITE Iperf
  Full Name:            Iperf & Iperf2 & Iperf.Iperf
  Source:               /var/lib/jenkins/workspace/test-iperf/test-robot/iperf.robot
  Start / End / Elapsed: 20190415 12:38:52.590 / 20190415 12:39:17.145 / 00:00:24.555
  Status:               3 critical test, 2 passed, 1 failed
                        3 test total, 2 passed, 1 failed
  + TEARDOWN SSHLibrary.Close All Connections
  + TEST Start Server IPERF
  − TEST Start Client IPERF TCP
    Full Name:            Iperf & Iperf2 & Iperf.Iperf.Start Client IPERF TCP
    Start / End / Elapsed: 20190415 12:38:53.972 / 20190415 12:39:05.476 / 00:00:11.504
    Status:               FAIL (critical)
    Message:              KO != OK
    + KEYWORD Open Connection to Client And Log In
    + KEYWORD ${output}, ${rc} = SSHLibrary.Execute Command ${CLIENT_TCP_START}, return_stdout=true, return_rc=true
    + KEYWORD BuiltIn.Should Be Equal As Integers ${rc}, 0
    + KEYWORD ${receiver} = String.Get Line ${output}, -3
    + KEYWORD @{rcv_list} = String.Split String ${receiver}
    + KEYWORD ${result} = BuiltIn.Set Variable If ${rcv_list[6]} > ${BANDWIDTH}, OK, ${rcv_list[6]} == ${BANDWIDTH}, OK, ${rcv_list[6]} < ${BANDWIDTH}, KO
    + KEYWORD BuiltIn.Set Suite Variable ${res}, ${result}
    − KEYWORD Validate Bandwidth KPI
      Start / End / Elapsed: 20190415 12:39:05.473 / 20190415 12:39:05.475 / 00:00:00.002
      − KEYWORD BuiltIn.Should Be Equal As Strings ${res}, OK
        Documentation:        Fails if objects are unequal after converting them to strings.
        Start / End / Elapsed: 20190415 12:39:05.474 / 20190415 12:39:05.475 / 00:00:00.001
        12:39:05.475  FAIL  KO != OK
```

Figure 37: 5G EVE Visualization GUI – Robot Framework Logs

4 Conclusion

This deliverable presents a high-level description of the first version of the Model-based Testing Methodology for 5G EVE and the high-level design and implementation aspects of the testing validation tools which become available to the sites and they are the main delivery target of D5.1. The testing/validation tools are available on the project site, in the following link: https://www.5g-eve.eu/end-to-end-facility/

The next steps regarding the Model-based Testing Methodology, are the description of the methodology for the generation of the low level test plan and the test procedures, as well as the application of the methodology to other use case as well. The final version of the methodology will be reported in deliverable D5.2. The next steps regarding the implementation of testing and validation testing tools is the implementation of the 5G EVE testing and validation framework which will be a universal framework integrated with all sites.

References

[1] https://iperf.fr/
[2] http://man7.org/linux/man-pages/man8/tc-netem.8.html
[3] https://ostinato.org/
[4] https://tcpreplay.appneta.com/
[5] https://jenkins.io/
[6] https://robotframework.org/
[7] https://github.com/adiralashiva8/robotframework-metrics