H2020-ICT-688712

Project: H2020-ICT-688712

Project Name:

5G Applications and Devices Benchmarking (TRIANGLE)

Deliverable D3.1

Progress report on the testing framework Rel 1

Date of delivery:	30/11/2016	Version:	1.0
Start date of Project:	01/01/2016	Duration:	36 months

Deliverable D3.1
Progress report on the testing framework Rel 1

Project Number:	ICT-688712
Project Name:	5G Applications and Devices Benchmarking
Project Acronym	TRIANGLE

Document Number:	ICT-688712-TRIANGLE/D3.1
Document Title:	Progress report on the testing framework Release 1
Lead beneficiary:	Universidad de Málaga
Editor(s):	Universidad de Málaga
Authors:	Keysight Technologies Belgium (Michael Dieudonne), Keysight Technologies Denmark (Andrea Cattoni, German Corrales Madueño), Universidad de Malaga (Alberto Salmerón, Almudena Díaz, Pedro Merino, Cesar A. García, Álvaro Rios, Álvaro Martín), Redzinc Services Limited (Jeanne Caffrey, Donal Morris, Ricardo Figueiredo, Terry O'Callaghan, Pilar Rodríguez), AT4 wireless S.A.U (Carlos Cárdenas, Janie Baños, Oscar Castañeda), Quamotion (Frederik Carlier, Bart Saint Germain)
Dissemination Level:	PU
Contractual Date of Delivery:	30/11/2016
Work Package Leader:	Universidad de Málaga
Status:	Final
Version:	1.0
File Name:	TRIANGLE_Deliverable_D3.1-FINAL

Abstract

This deliverable provides the description of the first release of the TRIANGLE testbed. It introduces the architecture, the workflow and the initial deployment of the testbed. In addition, a detailed description of all the components integrated in the testbed is provided. Moreover, for the sake of clarity, the deliverable describes a preliminary test execution with several of the components deployed in the testbed.

Keywords

Architecture, workflow, deployment, orchestration, test case, portal, measurements tools, RAN, EPC, SDN,EUs

Executive summary

This document is the first deliverable of WP3. WP3 is responsible for the development of the testing framework for Triangle. The testing framework covers all the software, and the coordination/sequencing that control & connects to the test infrastructure. It is in charge of handling and transforming the end user test requests into actionable steps for the software and hardware components of the testbed.

The document describes the initial architecture, the workflow and the deployment of the Triangle testbed. Also the deliverable provides a complete description of the features and the control interface of each one of the components of the testbed. The content of this document will be updated in D3.2, which will introduce advances in the architecture and the features provided by testbed.

The focus of the document is to provide a clear understanding of how the testbed is being designed to deliver testing and certification services to app developers, device makers and researchers. This first release is centred on the design of how the individual components will be coordinated, the design of the web portal for end-users, and how the measurements are collected. The document also includes the results of a preliminary test executed in the testbed to provide a clearer view of the type of testing and results that can be obtained.

We expect to have the first version of the testbed completely deployed and tested by the end of January 2017. The experiments and extensions selected in the first open call are planned to start in Q1 2017. These experiments and extensions will enable us to tune some aspects of the testbed such as the usability, the reporting, and its scalability regarding future extensions.

Contents

List of Figures

List of Tables

List of Abbreviations (Editor:)

AUT	App Under Test
AP	Access Point
APNet	Antennas, Propagation and Radio Networking
BER	Bit Error Rate
BLER	Block Error Rate
BS	Base Station
CAPEX	CApital EXpenditure
CDMA	Code Division Multiple Access
CFO	Carrier Frequency Offset
CO	Confidential
CP	Cyclic Prefix
CR	Cognitive Radio
CRS	Cognitive Radio Systems
CSI	Channel State Information
CSMA	Carrier Sense Multiple Access
C2X	Car-to-Anything
D	Deliverables
DL	Downlink
D2D	Device-to-Device
DMRS	Demodulation reference signal
DRX	Discontinuous Reception
DTX	Discontinuous Transmission
DUT	Device Under Test
EIRP	Effective Isotropic Radiated Power
EIT	European Institute for Innovation and Technology
E2E	End-to-End
EVM	Error Vector Magnitude
FDD	Frequency Division Duplex
FD-MIMO	Full-Dimension MIMO
FEC	Forward Error Correction
FR	Frequency Response
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications

HARQ	Hybrid Automatic Repeat Request
ICI	Inter-Carrrier Interference
ICT	Information and Communications Technology
IEEE	Institute of Electrical and Electronics Engineers
IMT	International Mobile Communications
IP	Intellectual Property
IPR	Intellectual Property Rights
IR	Internal report
ITU	International Telecommunication Union
ITU-R	International Telecommunication Union-Radio
KPI	Key Performance Indicator
LAN	Local Area Network
LOS	Line of Sight
LTE	Long Term Evolution
LTE-A	Long Term Evolution-Advanced
L2S	Link to System
M	Milestones
Mbps	megabits per second
Mo	Month
MA	Multiple Access
MAC	Medium-access Control
MGT	Management
MIMO	Multiple-Input Multiple-Output
MMC	Massive Machine Communication
M2M	Machine to Machine
MSE	Mean Squared Error
NLOS	Nonline of Sight
OFDM	Orthogonal Frequency Division Multiplexing
OPEX	Operational Expenditure
PA	Power Amplifier
PAPR	Peak-to-Average-Power-Ratio

PC	Project Coordinator
PHY	Physical Layer
PU	Public
QAM	Quadrature Amplitude Modulation
QAP	Quality Assurance Plan
QMR	Quarterly Management reports
QoE	quality of experience
QoS	Quality of Service
RACH	Random Access Channel
RAN	Radio Access Network
RAT	Radio Access Technology
RF	Radio Frequency
R&D	Research and Development
RRM	Radio Resource Management
RTD	Research and Technological Development
RTT	Round Trip Time
SDR	Software Defined Radio
SINR	Signal to Interference and Noise Ratio
SRS	Sounding Reference Signal
T	Task
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TRX	Transmitter
TTI	Transmission Time Interval
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System
USRP	Universal Software Radio Peripheral
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-anything
WCDMA	Wide Code Division Multiple Access
WLAN	Wireless Local Area Network
WP	Work Package
WPAN	Wireless Personal Area Networks

1 Introduction

1.1 Why Triangle

According to Cisco [1], global mobile data traffic grew 74 percent in 2015, reaching 3.7 exabytes per month at the end of 2015, up from 2.1 exabyte per month at the end of 2014. In addition, mobile data traffic has grown 4,000-fold over the past 10 years and almost 400-million-fold over the past 15 years and the fourth-generation (4G) traffic exceeded third-generation (3G) traffic for the first time in 2015.

The fast pace of the standardization process adds more complexity to this picture. In current network deployments, while evolving towards 5G, 4G LTE coexists with 2G and 3G technologies. In conjunction with the set of new features added in each 3GPP release, the potential combinations of network configurations grow exponentially for operators.

Despite of that, mobile subscribers are increasingly focused on applications and keep demanding high levels of quality everywhere. Unfortunately, it is not unusual to find performance and connectivity problems that hugely impact the user experience.

In this context, the primary objective of the Triangle project is to promote the testing and benchmarking of mobile applications and devices in Europe as the industry moves towards 5G. It also provides a pathway towards certification of qualified mobile developments in Europe.

Previous deliverables have identified the requirements to provide adequate testing means to increase the quality and reliability of mobile communications and applications in multiple domains. These requirements have shaped the Triangle testbed, which aims at providing an end-to-end testing framework for mobile devices and applications. This report describes the first release of Triangle testbed, introducing its high-level architecture, as well as the components that have been integrated into it.

1.2 The Triangle testbed and its users

Figure 1 provides an overview of the testbed with emphasis on the testing workflow.

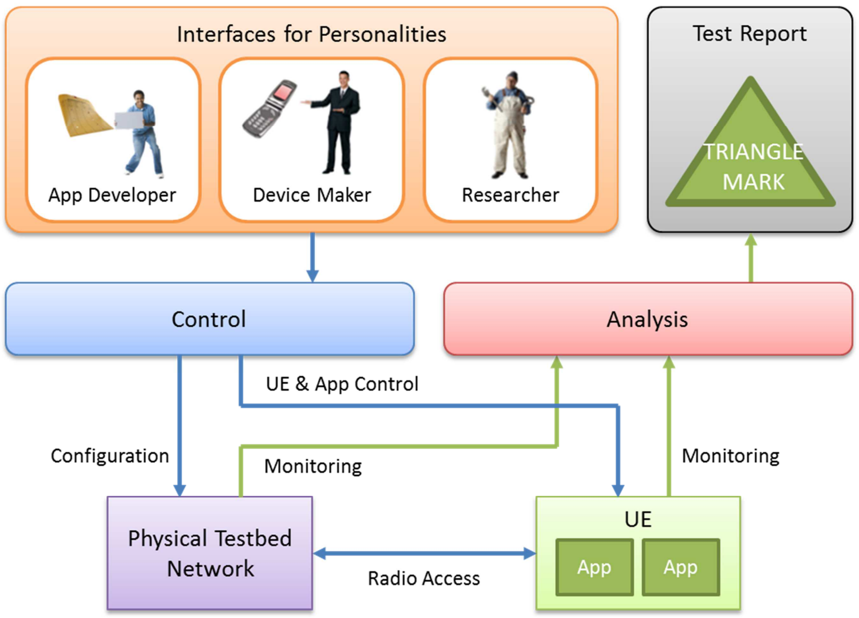


Figure 1 Overview of the Triangle testing workflow

The point of entry to the testbed is the interfaces offered to the end users. These interfaces are being designed to adapt to the user's needs. At the same time the complexity of the testing is wrapped into high-level scenarios, which will prevent users from having to deal with the full set of configurable parameters. These high-level scenarios are based on 5G testing scenarios identified in Section 3 of Deliverable 2.1.

Based on this selection, the testbed will configure the physical components, and schedule the execution of the tests and the collection of measurements required to check the performance of the features of the application or device under test. The interaction with the apps under test will also be automated, to carry out the user behaviours that will be analysed.

As a result, the testbed will provide detailed reports to the user. These reports will be based on the information collected by the reporting and automation tools running on both the testbed and the device.

The testbed will support two types of test procedures: execution of custom tests and execution of certification tests. In both cases the user will obtain a report with the results of the tests executed, but in the later the testbed will provide the certification mark and the certification report. All mandatory and applicable test cases that must be executed in the certification procedure, and the results are compared with reference values, in order to obtain the Triangle mark.

In order to provide a more clear idea of the tests cases that can be executed in the Triangle testbed Section 11 describes a test example. The BlueEye wearable system will be treated as an alpha customer of the Triangle testbed for IoT devices. Section 12 provides an initial proof of concept for an IoT device.

1.3 Testbed high-level architecture

Figure 2 shows a more detailed view of the main functional blocks that make up the Triangle testbed architecture. This architecture can be divided into several subsystems, whose role will be introduced briefly in the rest of this section. The figure also includes a note with the section in which each component is described in more detail.

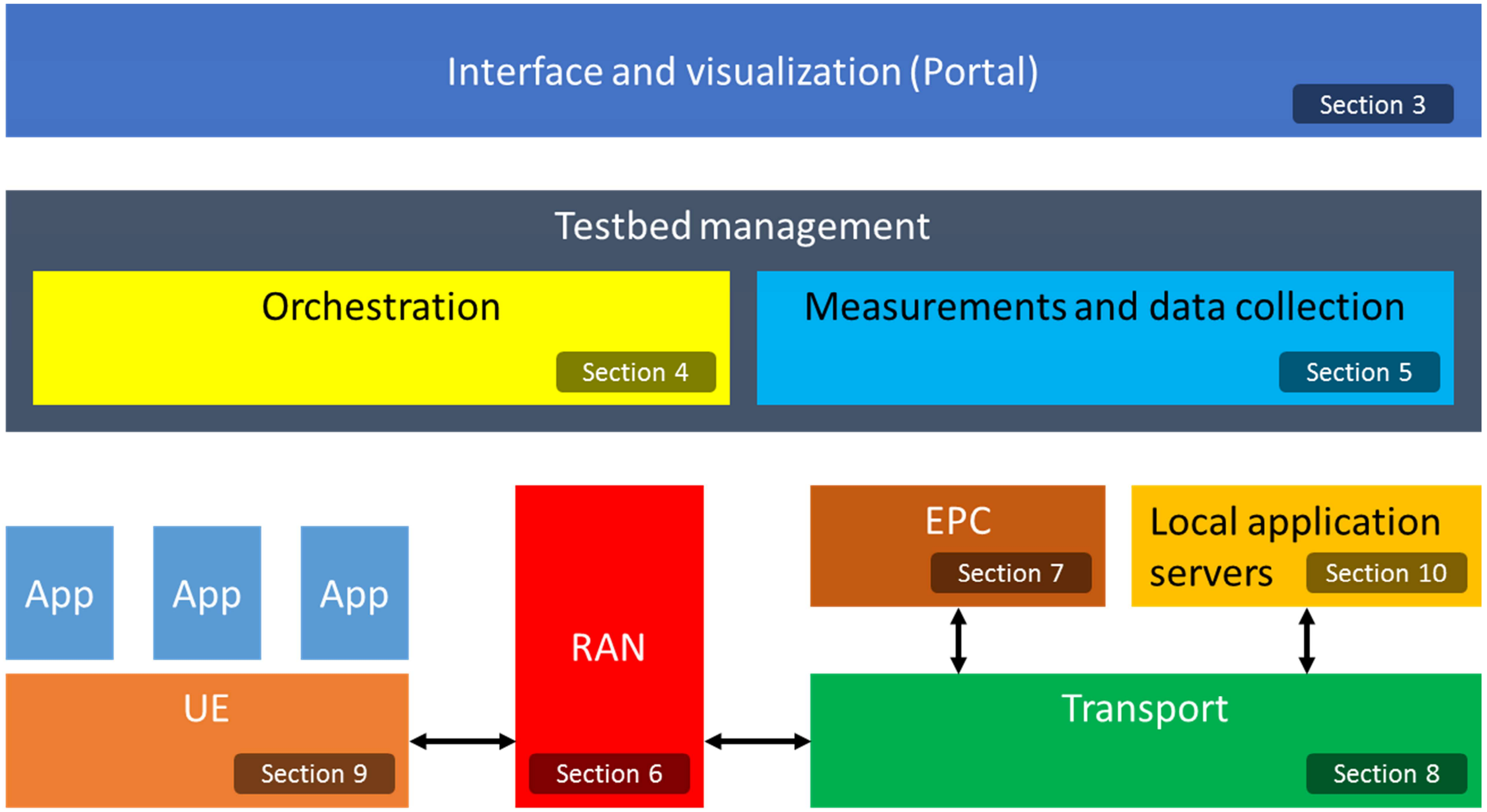


Figure 2 High-level architecture of the Triangle testbed

1.3.1 Interface and visualization (Portal)

The Triangle portal is a user-friendly interface for remote interaction with the testbed. It provides a view of the testbed that is adequate for each user profile, hiding unnecessary complexity. The main purpose of the testbed portal is preparing and running tests, and later reviewing the results.

Tests will be configured based on the selections performed by the users, e.g. which app to test, on which device, and on which high-level scenario. These inputs given by the user are processed and transformed into inputs for the different components of the underlying architecture.

A high-level scenario is an understandable term of the network conditions which will be configured during the test case to reproduce conditions experimented in the high-level scenario selected. In other word, a high-level scenario is an abstraction of similar network configurations, e.g. “Vehicular” is a high-level scenario which compromises different configurations of the speed: 30 km/h, 60 km/h, 90 km/h and 120 km/h.

The portal will store all the campaigns and other user provided data, as well as the results obtained from the testbed, so that test case results are completely traceable to their configuration, and can be repeated if needed.

The initial version of the interfaces implemented for the app developer profile are described in Section 3.

1.3.2 Orchestration

To run a test case, all components must be controllable by an orchestrator, which must coordinate their configuration and execution. In the Triangle testbed, the Test Automation Platform (TAP), from Keysight, will serve as coordinator responsible for configuring and running the tests. Each testbed component is controlled through a TAP driver, which serves as bridge between the TAP engine and the actual component interface. Some of these drivers are being developed with the testbed.

The configuration of the different elements of the testbed network will be determined by the high-level scenarios selected by the users. The testbed will translate these high-level scenarios into the specific configurations that TAP will apply to each network component.

An integral part of testing apps is automating their execution, i.e. simulating the interactions of a user with the app. Quamotion automation tools provide the means to create sequences of user actions, and then replaying them on a testbed device.

To synchronize radio access and power consumption measurements, the orchestration components include a PTP based synchronization system.

These orchestration components are described in Section 4.

TAP will control the overall execution of each test case.

1.3.3 Measurement and data collection

A measurement is a value discovered by measuring, that corresponds to a property of something. To obtain measurements to compute the Triangle Mark and other reports, the testbed will provide several probes (both software and hardware) which will extract the required measurements. Software probes running on the UE include AT4 Agents and the TestelDroid tool from UMA. Triangle will also provide an instrumentation library for app developers, in order to provide additional measurements that cannot be extracted by other means. Hardware probes include a power analyzer connected to the UE to measure power consumption.

These measurements will be collected and analyzed in order to calculate the key performance indicators (KPIs) associated to the features provided by the apps or devices, e.g. video streaming or VoIP calls. To facilitate the aggregation of measurements and KPIs, all will be stored in a central OML server, which uses a PostgreSQL database server as backend.

All these measurement components are described in detail in Section 5.

1.3.4 RAN (Radio Access Network)

Radio access emulation plays a key role in the Triangle testbed. In the first release of the testbed, RAN will be provided by a UXM Wireless Test Set from Keysight, a flagship mobile network emulator that provides state of the art test features. Some of its key features for testing include flexible Inter Cell Interference Coordination (eICIC) schemes, WLAN offloading, IMS/End to End VoLTE communications between multiple devices, and battery drain performance with flexible network and sleep mode settings.

Only to UEs can be connected at the same time to the UXM. To connect more devices, e.g. all the reference devices used in the testbed, RF switches will be used.

The RAN technologies used in the first release of the testbed are described in Section 6. RAN technologies for future releases of the testbed, such as Wi-Fi access points and Small Cells, are described in Section 13.

1.3.5 EPC (Evolved Packet Core)

In order to provide an end-to-end system, we will integrate in the testbed a commercial EPC from Polaris Networks, which includes the main elements of a standard core network: MME, SGW, PGW, HSS, and PCRF. In addition, this EPC includes the EPDG and ANDSF components for dual connectivity scenarios. As an extension for this project, we are developing the S1 interface to interconnect the UXM with the EPC.

This EPC system is described in Section 7.

1.3.6 Transport

To emulate the transport network between the eNodeB and the EPC, we will use an SDN deployment that provides features such as traffic prioritization, separation of data and control plane traffic, and transparent mirroring of selected traffic flows. Moreover, the testbed will offer the possibility of integrating artificial impairments in the interfaces of the core network and the application servers.

Finally, the testbed includes an over-the-top-content enabler, the VPS Engine, which can be used by third-party applications to configure certain aspects of the SDN deployment and the EPC policies to request a specific quality of service (QoS).

The transport network components are described in Section 8.

1.3.7 App(s)

The Triangle testbed will be able to run apps from app developers in a set of reference devices, in order to test them. Support for running and automating apps is explained in Section 4, as part of the orchestration of the testbed.

On the other hand, the testbed will employ reference apps in order to test and measure the performance of devices under test. Some of these reference apps and their usage are described in Section 5.

1.3.8 UE (User Equipment and accessories)

As with apps, the testbed will allow both reference devices to test apps provided by users, and new devices that will be tested under the conditions set by the testbed. In both cases, devices have to be physically connected to the testbed. First, in order to preserve the radio conditions configured at the UXM Wireless Test Set, the RF connection must be conducted through cables. Second, to analyze properly power consumption, the device must be powered directly by the N6705B power analyzer.

In addition, the device should provide some control and automation interface that can be used from the testbed orchestration tools. For instance, in the case of Android, this means a USB connection to a testbed computer to support connection through the adb tool.

The integration of a set of reference Android devices is described in Section 9.

1.3.9 Local application servers

As part of some reference apps, the testbed will provide local application servers. Allocating these servers in the testbed will ensure that measurements from these reference apps will not be altered by the Internet connection outside of the reach of the testbed.

For the first release of the testbed, the installed local application servers include a VoIP server and a video streaming server, both introduced in Section 10.

1.4 Testbed workflow

Figure 3 provides the overall logic flow of the testbed from the top down.

The workflow of the testbed starts with the user defining, through the Web portal, the app/device under test, their features and user behaviours, and the selection of procedure: certification or custom test. For custom tests, the user will also have to specify the high-level scenarios in which he or she is interested. In this context, user behaviours are sets of actions that represent particular usage scenario of the application, such as login in or streaming a video. For instance, login into a particular application means pressing a button, entering a text in the username field, entering a text in the password field, and clicking the “Ok” button.

The testbed will transform these inputs into:

- Specific configurations of the network elements. For each configurable component, the testbed will select among a predefined pool of configuration files the ones that match the input given by the user. The resulting test plan to be executed will use the contents of this files to configure each component.
- User behaviours will be translated into a specific user flow control, which will be included in the test plan to automate the behaviour of the application during the test.
- KPIs associated to the features declared by the user. The test plan will take into account these KPIs to define the measurements that have to be collected during the test.

The orchestrator will conduct the execution of the test and the collection of the measurements, which will be stored in a common database for each test. The results will analyzed and post-processed to calculate the KPIs, generate the test reports and (when applicable) deliver a Triangle mark.

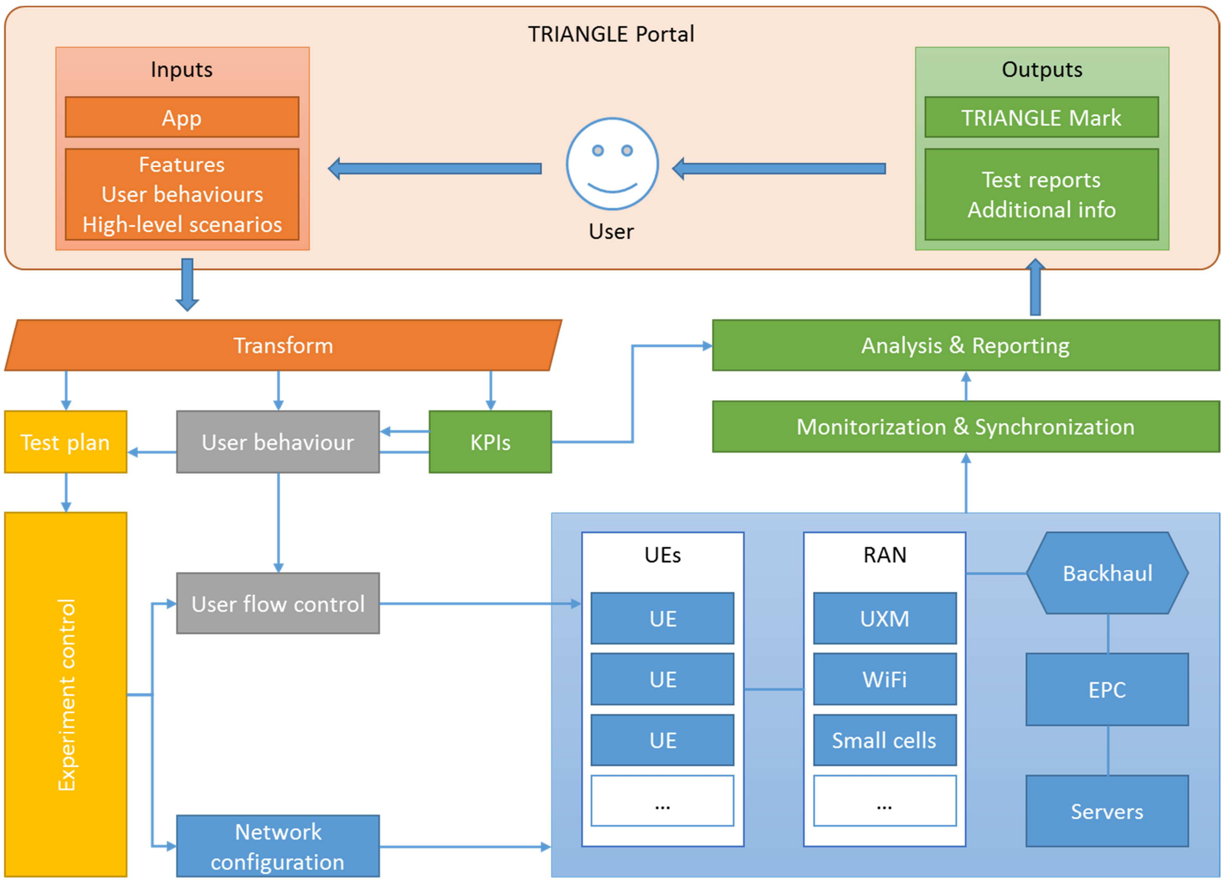


Figure 3 Triangle testbed high-level workflow

2 Triangle testbed architecture

The Triangle testbed is the set of components that allows the implementation of the Triangle Testing Framework. The testbed is composed of several interconnected hardware units, computers and virtual machines. All these components work together to provide the means to execute tests over apps or devices, and to provide test reports. While Figure 2 shows the high-level components of the testbed, this section how this components are connected, both physically and by their software interfaces.

2.1 Physical architecture

Figure 4 shows an overview of the physical interconnections between the testbed components.

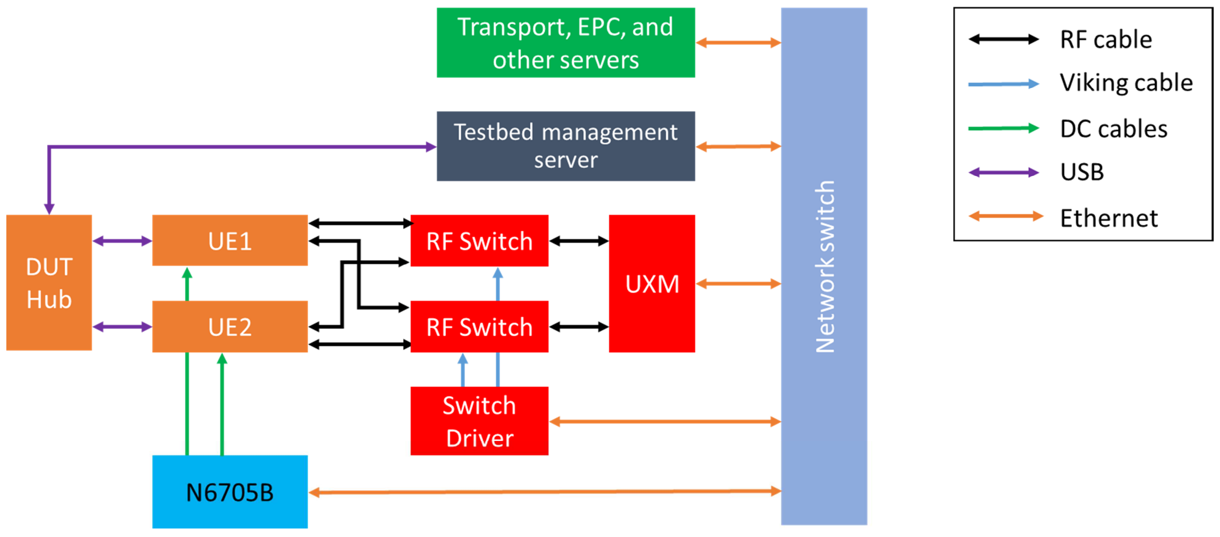


Figure 4 Testbed physical architecture

UEs (either reference devices for app testing, or a UE under test, when testing devices) are connected to the UXM mobile network emulator. The radio signal is not radiated; it is conducted through calibrated RF cables to the UE antenna connector. For testing purposes most UEs typically contain small antenna connectors which are normally hidden from the user. The UXM only support the interconnection of two UEs. To connect more devices to the same UXM, the testbed will use RF switches, controlled by a switch driver. The switches will be placed in the RF connection between the UEs and the UXM, and the switch driver will select which RF connections will actually be routed to the UXM.

The UXM emulates all the network signalling and physical signals, as well the MIMO radio channel. All the protocol layers in the emulated network operate realistically as defined in the 3GPP test specifications and can be configured. Moreover, for testing purposes, the UXM instrument provides a number of additional useful capabilities, such as a downlink channel emulator, detailed logging and friendly control.

The battery pins of a UE are connected to the DC power analyzer. This allows both the control of the input voltage into the phone and to measure the instantaneous current drawn from N6705B power analyzer. The N6705B power analyzer supports up to four devices connected at the same time.

The UE will also be connected through a USB data cable to a server running the testbed automation software. Again, to support several devices connected to the same server and switch between them, the UE USB data cables will actually be connected to a DUT HUB, which in turn is connected to the server by a single USB cable.

Finally, the following elements are connected via Ethernet in a local network: UXM, N6705B, switch driver, orchestration server, EPC, transport, and additional services.

2.2 Software architecture

Figure 5 shows the software interfaces between the testbed orchestrator and the controlled components, and with the testbed Portal.

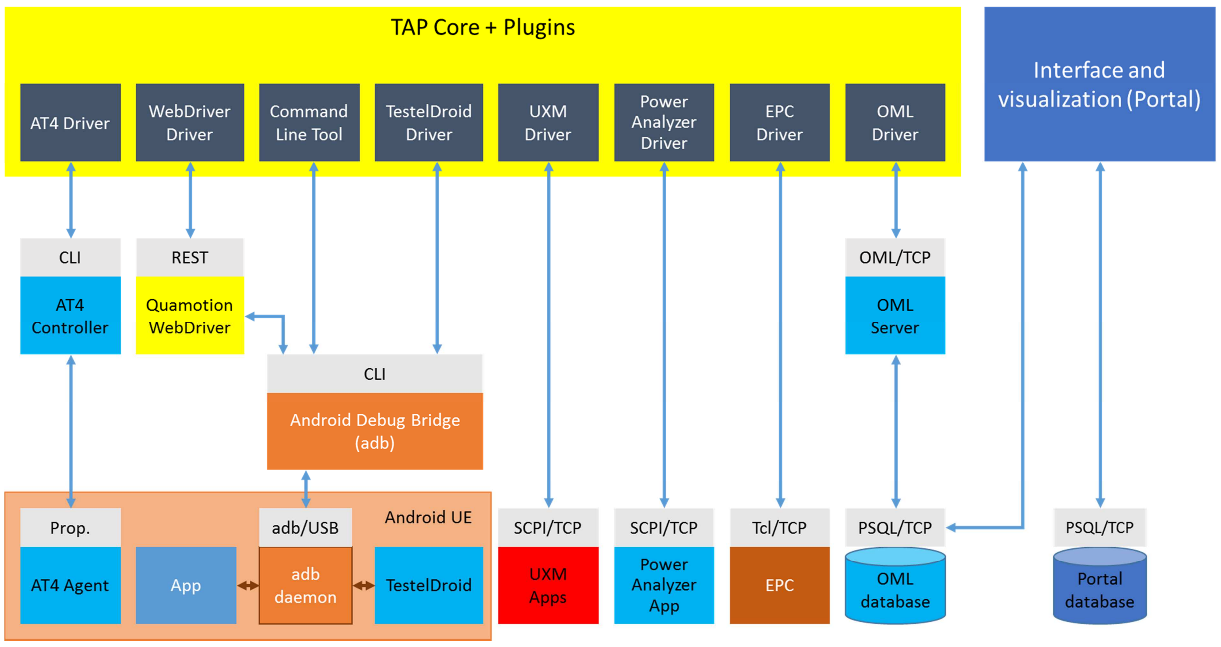


Figure 5 Testbed software architecture

At the top of the figure lies TAP, the orchestrator of the testbed components. TAP controls all the components of the testbed through appropriate TAP drivers provided by TAP plugins. The TAP core also comes with a set of standard plugins.

Many components offer a SCPI interface to receive commands. TAP provides support for writing drivers for SCPI-based components, which facilitates the work of adding the required support. This is the case of the drivers for the UXM and Power Analyzer apps running on their corresponding hardware units. In both cases, the SCPI commands will be delivered through a TCP connection.

For the first release of the testbed, some of the software components will be managed through their command-line interfaces (CLI, in the figure). For instance, the AT4 Controller will be invoked using the command-line tool it provides. The driver for the AT4 Controller will be a thin wrapper around the command-line interface.

For Android apps, the adb command-line tool is a fundamental component. Adb can be used to send commands to apps running on the UEs, or automate certain actions such as switching airplane mode on and off to force the attachment of the UE to the base station. Some of the UE automation tools that are part of the testbed, such as Quamotion WebDriver, use adb to perform their function.

The Quamotion WebDriver provides a REST API to manage the apps on an Android device, and perform user actions, such as tapping or swiping. These commands will be delivered to the Android UE using adb. TestelDroid is also managed through adb.

For the first release of the testbed, the EPC will be controlled through a fixed set of Tcl scripts that use a Tcl scripting API provided by the EPC components emulators. The TAP driver will execute these scripts, which then will send the appropriate commands to the ECP through the Tcl API.

The measurements from all the tools will be collected and send to a central OML server, which uses a custom OML protocol. While some tools may send measurements directly to the OML server, the TAP orchestrator will use a driver that will allow sending measurements to the OML server from TAP, in two ways. First, for tools that generate results in CSV files, the driver will collect these files and send them as measurements to the OML server. Second, the driver will implement the standard TAP mechanism for handling results from drivers, so that drivers which are already well integrated with TAP can publish them to the OML server without additional work.

2.2.1 Software deployment

Regarding the deployment, software will be split between Windows and Linux machines (as required by each tool), and virtualized when possible. For the first release of the testbed, the current deployment is split into a single Windows machine, and several Linux virtual machines (VMs). Table 1 includes an overview of the installed software components.

This windows computer will contain the TAP platform, the adb server that can talk directly to the (Android OS) UE and the AT4 controller which manage the AT4 agents installed at UE. Other tools that use adb to relay commands to the UE, such as the Quamotion WebDriver, will need to be installed in the same computer, or be configured to use adb remotely.

Table 1 Software deployment in the testbed

Software component	**OS**	**Notes**
Test Automation Platform	Windows	Main computer
AT4 performance tool controller	Windows	Main computer
AT4 agent	Android / iOS	UE
TestelDroid	Android	UE
Quamotion WebDriver	Windows/Linux	Helper instance in Linux as part of Portal
Testbed Portal	Linux	VM, automated recreation
OML / PostgreSQL	Linux	VM, automated recreation
Local application servers	Linux	VM, automated recreation
adb server	Windows	Main computer
EPC	Linux	Multiple VMs

Linux VMs have been defined using Vagrant [13], which allows recreating them from scratch and updating their contents easily. Vagrant uses a mixed declarative and imperative approach to describe the contents of a VM, including the actions that must be performed to set up any required software that will run on the VM. The OML and the local servers are installed in these VMs. The EPC is distributed across a set of Linux VMs.

The Portal is deployed in a separate Linux VM. This machine hosts the web server that runs the Portal, as well as the database that is used as backend.

3 Interface and visualization (Portal)

The Triangle testbed portal (Portal, in the rest of this section) is the website that end users will use in order to run tests and review their results, with a user-friendly interface tailored to their needs. Triangle end user can be app developers, device makers, researchers, etc. End users may want to use the Triangle testbed to run available test cases, develop and run their own test cases or run the test cases required to obtain the Triangle mark. For the first release of the testbed, the focus has been set on supporting the app developer.

3.1 Organization

The organization and contents of the Portal have been designed as low fidelity mockups, i.e., mockups that focus on the content of each page, rather than providing a complete design. The data model of the entities managed by the portal have been defined with text and UML diagrams. Since the Portal will make extensive use of a SQL database as its backend, the expected tables have also been defined prior to the implementation.

The main sections of the Portal (for an app developer user) are:

- Login: a standard login page, integrated with LDAP. For the initial releases of the Portal, it is expected that login credentials will be provided directly to users, with no sign up page.
- Dashboard: the main page that users will see when they log in into the Portal. It contains a summary of the activity of the user, e.g. apps uploaded or tests performed.
- Apps: app developers can manage their apps in this section. They can upload new app files, which will be recognized and categorized automatically. For each app, they can see the test campaigns (i.e., the group of test cases selected by the user) in which it has been used.
- Booking: testbed booking is managed through the Odoo CRM software. However, users will be able to see their booked timeslots in the Portal.
- Test Campaigns: Users create test campaigns to test their apps in a particular device and scenario. A test campaign is a set of test cases selected by the user to be executed from one or several test specifications.
- One specific test campaign: is the certification test campaign. The certification test campaign is the set of test cases required to obtain the Triangle mark. The set of test cases in the certification test campaign is automatically generated based on the app characteristics.
- Help: information about the usage of the Portal.

These sections are shown in Figure 23.

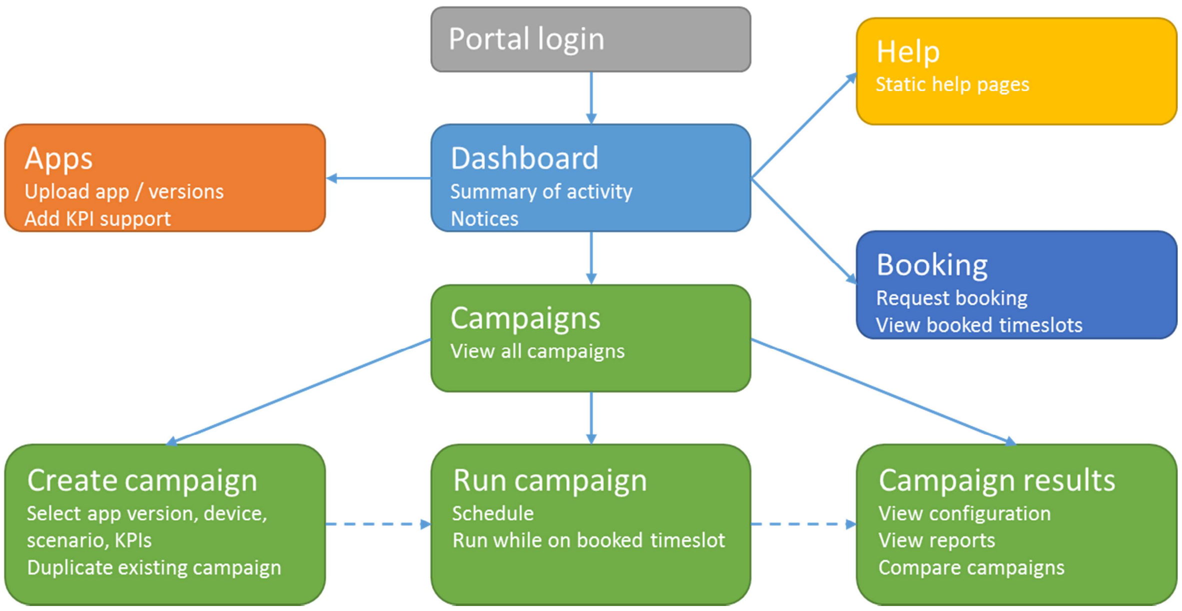


Figure 6 Overview of Portal organization for app developers

3.2 Data

The Portal has to store data to support the features outlined above. The main purpose of the Portal is to prepare and run tests. The main entities managed by the Portal and their relationship are shown in Figure 7. They are described in the remainder of this section. The names of the entities managed by the portal are written in uppercase, to highlight them.

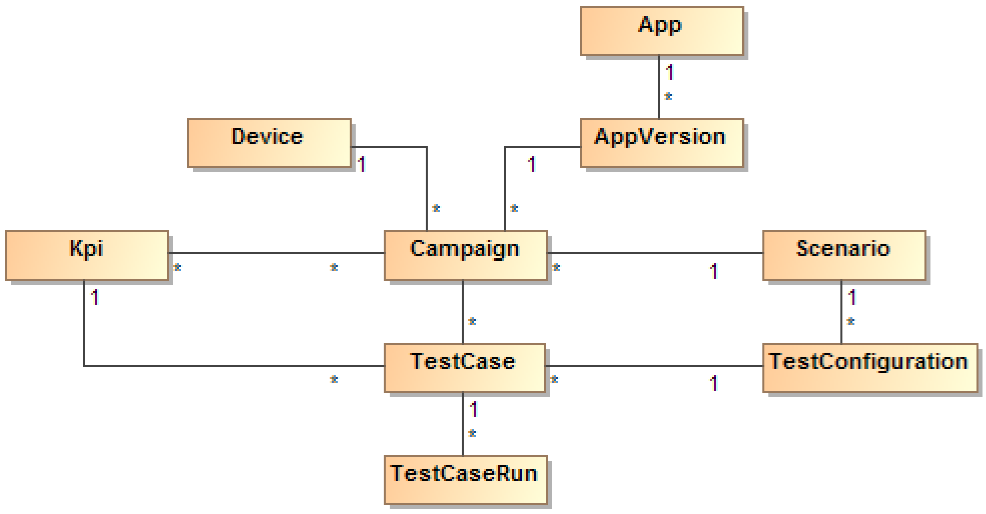


Figure 7 Main entities for app developers in the Triangle Portal

App developers upload apps to the Portal, to run tests on the reference devices provided by the testbed. The app developer must select the features implemented by the app, from a predefined list of features. Once the app has been upload the developer can run test cases. The developer can later upload several app versions (e.g. APK files for Android) of the same app, and compare the results obtained in different test cases. Different apps from the same app developer will be isolated, i.e., the user will not be able to compare results from test across different Apps.

Apps will be tested according to the test cases selected in the test campaign. The test cases that build the test campaign are automatically selected based on the features of the app, previously selected by the app developer. The developer can modify the list of test cases in the test campaign or can change the test cases which are provided in the Triangle testbed to run its own test cases. The test case automation controls instruments, devices, etc., captures measurements and compute KPIs. The measurements and the KPIs are made available to the users of the Triangle testbed. Furthermore, measurements and KPIs are used also to compute certain metrics. A metric is a categorization of the KPIs from a user satisfaction perspective. The metrics are used to provide the assessment before the Triangle mark can be granted.

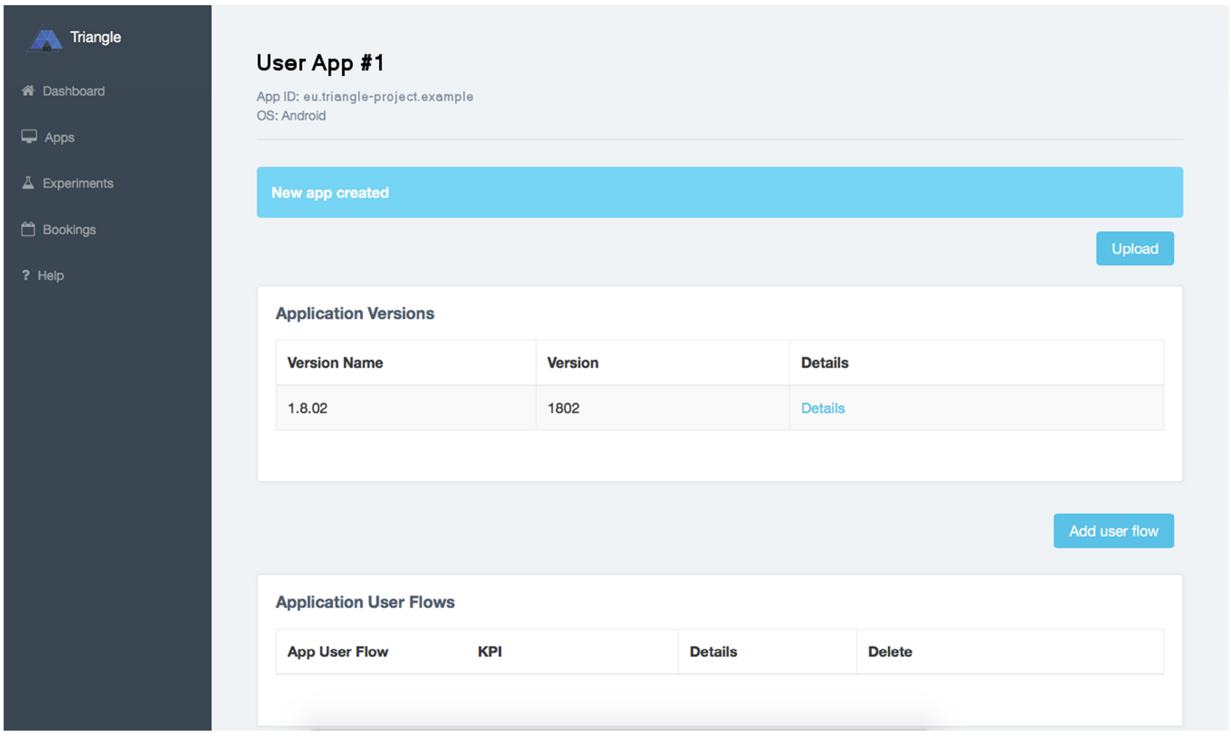
To run test cases with different app versions, app developers will create different test campaigns. For each test campaign, the app developer can select the device on which to run the test cases, the high-level scenario from a fixed set and, optionally, a subset of the applicable KPIs. A single high-level scenario abstracts several concrete test configurations (e.g., “Urban” includes “Pedestrian” and “Driving”), with the actual network configuration that will be applied.

Each test campaign includes a list of test cases. A test case is a sequence of actions required to achieve a specific test purpose. Each time the same test campaign is executed, all its test cases will be executed. Each one will produce a new set of results (test case run, i.e. the results and metadata of the execution).

3.3 Implementation

The Portal is being implemented using the Ruby on Rails framework [11]. For the backend, the Portal uses a PostgreSQL database (different from the ones used by the OML server or TAP). For the frontend, the Portal uses the Bootstrap framework [12]. The Portal follows the organization outlined above. The data stored in the backend is structured as described in the previous subsection. The database tables currently defined are listed in Appendix 4.

The following figures show part of the current implementation of the Portal. Figure 8 shows the main page of an app uploaded by an app developer, with all the uploaded versions, and the app user flows defined.



User App #1

App ID: eu.triangle-project.example
OS: Android

New app created

Application Versions

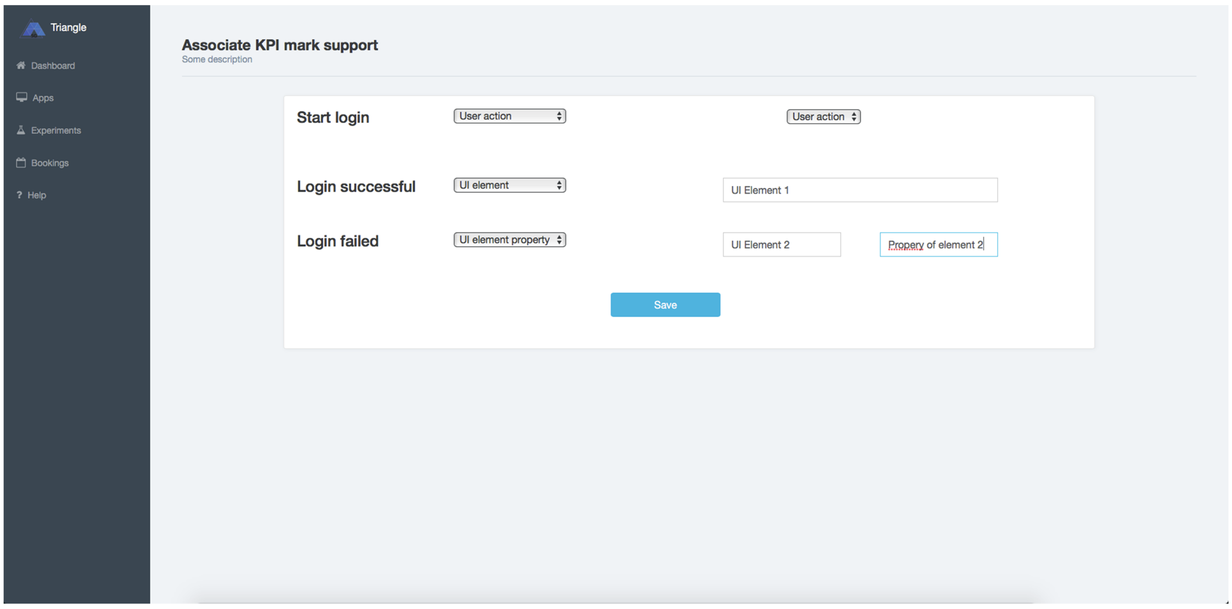
Version Name	Version	Details
1.8.02	1802	Details

Application User Flows

App User Flow	KPI	Details	Delete

Figure 8 Triangle Portal: app details page

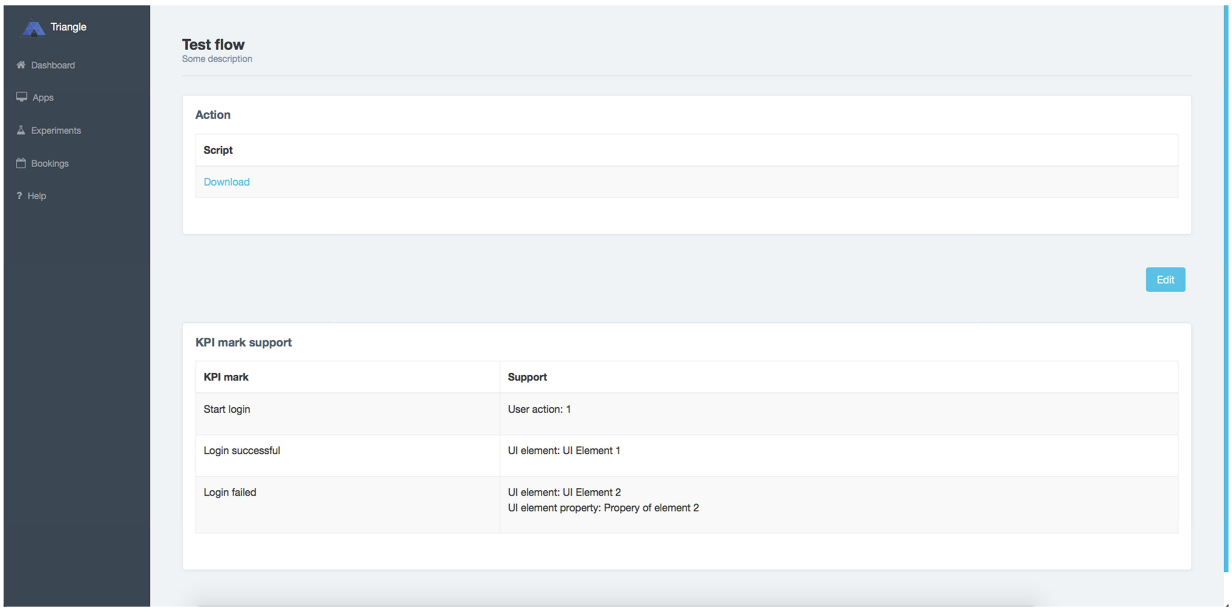
The creation of an app user flow is partially shown in Figure 9, where the testbed user declares how each measurement point for a given KPI will be provided. Figure 10 shows the result of creating an app user flow.



Associate KPI mark support
Some description

Start login	User action		User action
Login successful	UI element	UI Element 1	
Login failed	UI element property	UI Element 2	Propery of element 2

Figure 9 Triangle Portal: app user flow creation



Test flow
Some description

Action

Script
Download

KPI mark support

KPI mark	Support
Start login	User action: 1
Login successful	UI element: UI Element 1
Login failed	UI element: UI Element 2 UI element property: Propery of element 2

Figure 10 Triangle Portal: app user flow details page

4 Orchestration

4.1 Test Automation Platform

This section provides the features of the control tool selected to carry out the orchestration of the testbed. This orchestration is needed to coordinate the different components of the testbed. The accuracy provided by OMF is not enough to synchronize the executions of the network behaviours (such as handovers), app actions and the collection of the measurements. In other words, OMF is not well-suited for time sensitive operations.

TAP provides flexible and extensible test sequence and test plan creation. TAP is a Microsoft .NET-based application that can be used stand-alone or in combination with higher-level test executive software environments developed by Keysight. Leveraging C# and Microsoft Visual Studio, TAP is a platform upon which it is possible to build tests solutions.

TAP plays a key role in Triangle as it allows controlling and automating all the instruments present in the testbed. Included with Keysight TAP is the core sequencing engine, tools and plug-ins to minimize test system development time and test execution speed.

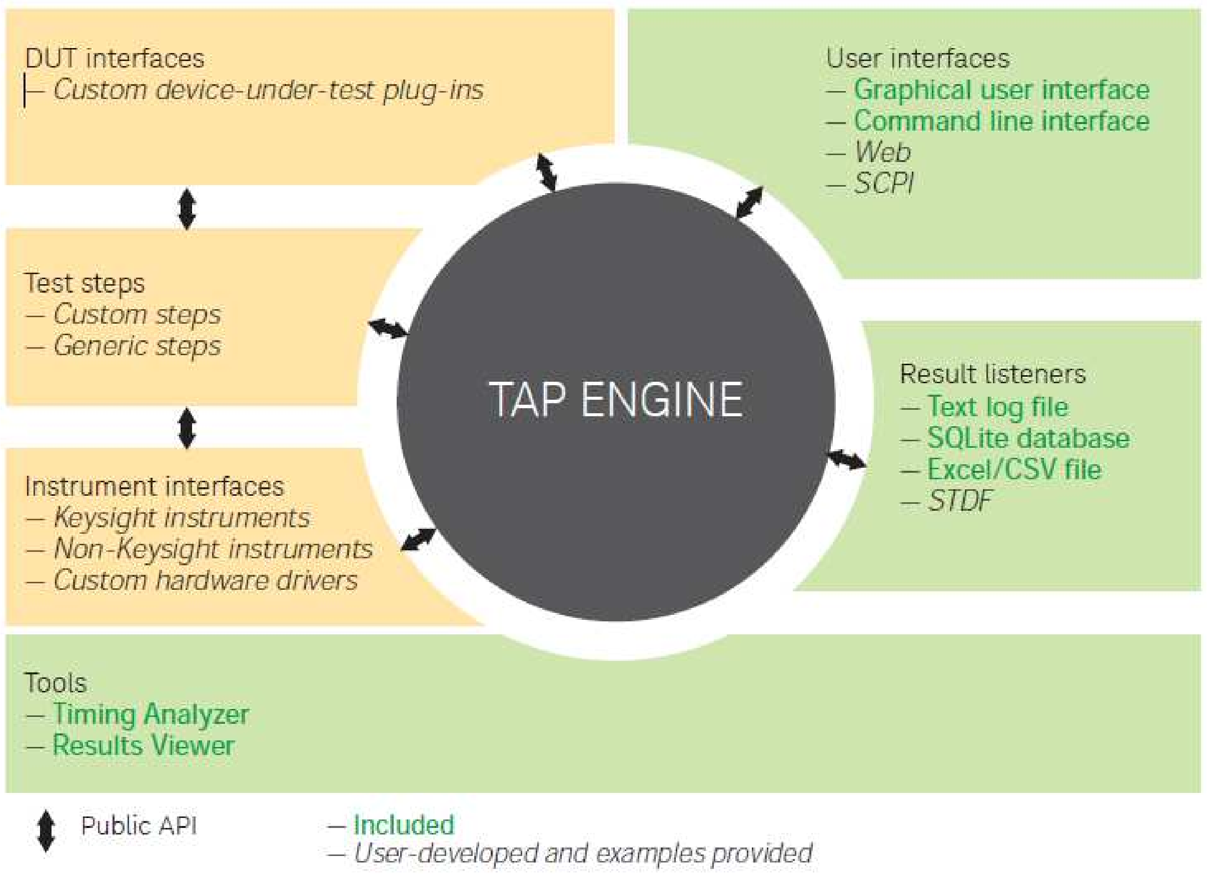
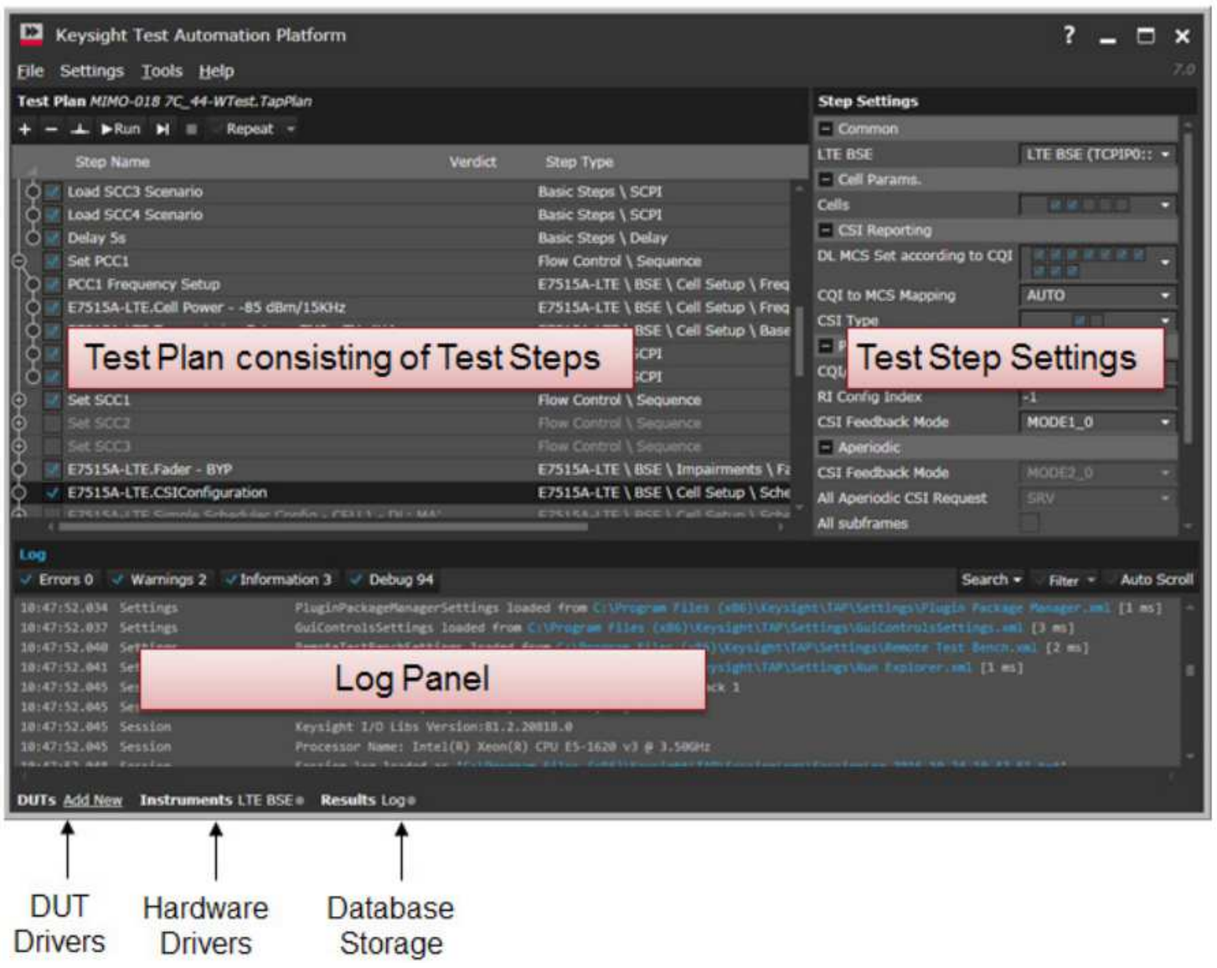


Figure 11 TAP architecture, showing core sequencing software engine.

4.1.1 Core Sequencing Engine

The Core Sequencing Engine is the “heart” of TAP, designed for speed-optimized test step execution. Tests (called test plans in TAP terminology) can include simple flow operations such as IF and LOOP. Complex hardware setups and parallel tests steps are also supported. The graphical interface of the core engine is shown in Figure 12.



Step Name	Verdict	Step Type
Load SCC3 Scenario		Basic Steps \ SCPI
Load SCC4 Scenario		Basic Steps \ SCPI
Delay 5s		Basic Steps \ Delay
Set PCC1		Flow Control \ Sequence
PCC1 Frequency Setup		E7515A-LTE \ BSE \ Cell Setup \ Freq
E7515A-LTE.Cell Power - -85 dBm/15KHz		E7515A-LTE \ BSE \ Cell Setup \ Freq
		BSE \ Cell Setup \ Base
		CPI
		CPI
Set SCC1		Flow Control \ Sequence
Set SCC2		Flow Control \ Sequence
Set SCC3		Flow Control \ Sequence
E7515A-LTE.Fader - BYP		E7515A-LTE \ BSE \ Impairments \ Fa
E7515A-LTE.CSIConfiguration		E7515A-LTE \ BSE \ Cell Setup \ Sche

Step Settings

Common	
LTE BSE	LTE BSE (TCPIP0::
Cell Params.	
Cells	
CSI Reporting	
DL MCS Set according to CQI	
CQI to MCS Mapping	AUTO
CSI Type	
RI Config Index	-1
CSI Feedback Mode	MODE1_0
Aperiodic	
CSI Feedback Mode	MODE2_0
All Aperiodic CSI Request	SRV
All subframes	

Log
Errors 0 Warnings 2 Information 3 Debug 94 Search Filter Auto Scroll

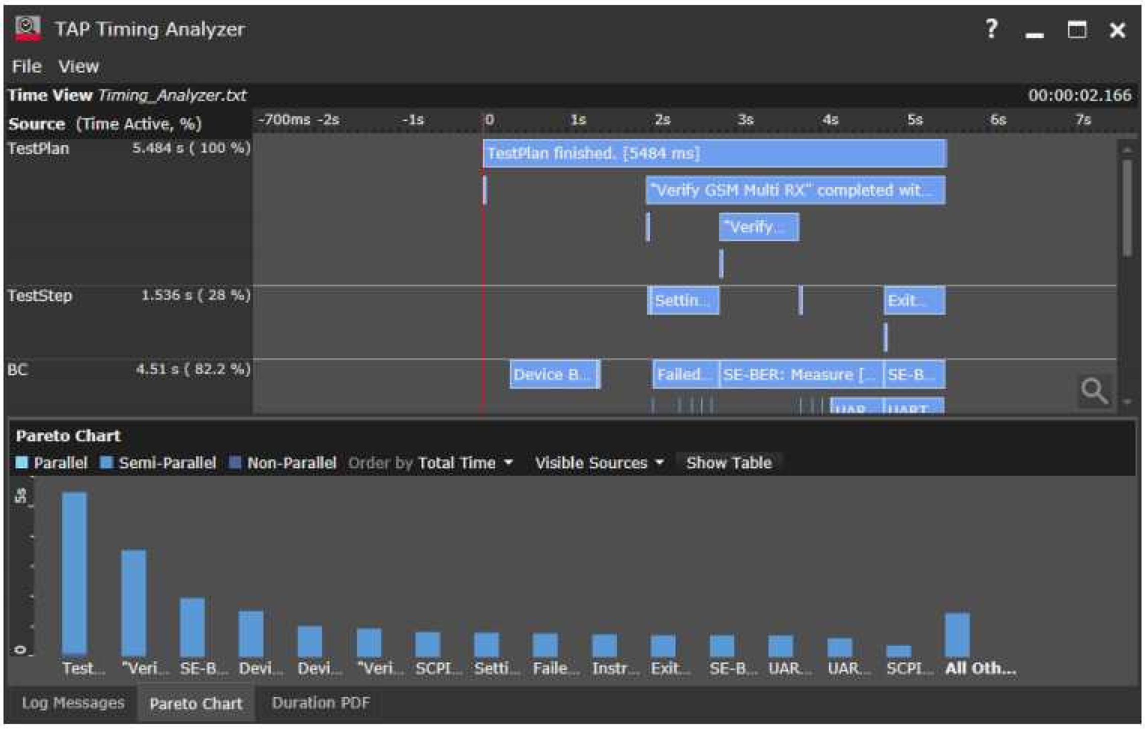
```
10:47:52.034  Settings    PluginPackageManagerSettings loaded from C:\Program Files (x86)\Keysight\TAP\Settings\Plugin Package Manager.xml [1 ms]
10:47:52.037  Settings    GuiControlsSettings loaded from C:\Program Files (x86)\Keysight\TAP\Settings\GuiControlsSettings.xml [3 ms]
10:47:52.040  Set                                                  )\Keysight\TAP\Settings\Remote Test Bench.xml [2 ms]
10:47:52.041  Set                                                  ysight\TAP\Settings\Run Explorer.xml [1 ms]
10:47:52.045  Se                                                   ck 1
10:47:52.045  Se
10:47:52.045  Session     Keysight I/O Libs Version:81.2.20818.0
10:47:52.045  Session     Processor Name: Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz
```

DUTs Add New Instruments LTE BSE Results Log

Figure 12 TAP’s Core Sequencing Engine and GUI

4.1.2 Timing Analyzer

TAP’s Timing Analyzer Tool provides insights into optimizing the overall test execution speed. It also allows visualizing the overall and in depth test execution time to see how much time each test step requires. The Timing Analyzer Tool is shown in Figure 13.


TAP Timing Analyzer

Time View *Timing_Analyzer.txt* 00:00:02.166

Source (Time Active, %)	
TestPlan	5.484 s (100 %)
TestStep	1.536 s (28 %)
BC	4.51 s (82.2 %)

Figure 13 TAP's TAP's Timing Analyzer test step's execution information

4.1.3 Results Viewer

Each time a TAP test plan is executed the results are stored in a database in TAP which can be graphed and visualized using the TAP Results Viewer. Multiple data sets can be viewed to quickly compare results across different test runs (i.e., several executions of the same test). The Run Explorer helps manage test plan data, recall old test plans, merge and compare test log timings, compare test plan settings, search for specific test results, and plot them using the Results Viewer.

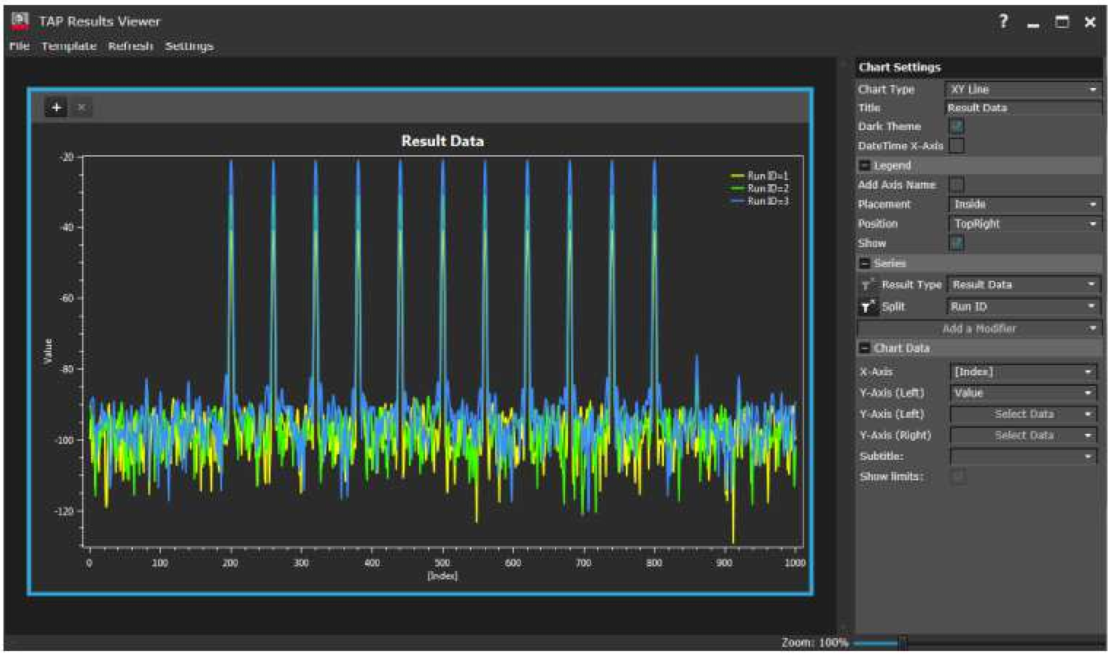

TAP Results Viewer

Chart Settings

Chart Type	XY Line
Title	Result Data
Dark Theme	✓
DateTime X-Axis	
Legend	
Add Axis Name	
Placement	Inside
Position	TopRight
Show	✓
Series	
Result Type	Result Data
Split	Run ID
Add a Modifier	
Chart Data	
X-Axis	[Index]
Y-Axis (Left)	Value
Y-Axis (Left)	Select Data
Y-Axis (Right)	Select Data
Subtitle:	
Show limits:	

Figure 14 TAP Result Viewer provides a quick and flexible test run data visualization.

4.2 Quamotion automation tools

4.2.1 WebDriver

The Quamotion WebDriver is able to automate user actions on iOS and Android applications whether they are native, hybrid of fully web based. The Quamotion WebDriver handles the full lifecycle of app usages (installation, starting and automation) without any manual interaction. The only requirements are a valid application package (apk or ipa file) and in case of iOS a DeveloperProfile [6] and the iOS DeveloperDisks.

The design of Quamotion WebDriver follows the specifications of the W3C WebDriver specification [7]. In origin, the WebDriver provides a platform- and language-neutral protocol to remotely instruct the behaviour of web browsers.

The Quamotion WebDriver is an extension on the WebDriver specification and adds specific support to manage mobile devices and mobile applications. The Quamotion WebDriver is able to instruct the behaviour of mobile application similar to instructing the behaviour of web browsers.

The following instructions are added on top of the standard WebDriver instructions in order to support test automation for mobile applications.

Table 2 Command reference for mobile extensions to the WebDriver

HTTP method	**Path**	**Summary**
POST	/session	Creates a new session, starting the application on the device.
PUT	/quamotion/developercenter/profile	Creates a new Apple developer profile.
POST	/session/:sessionId/quamotion/elementByCoordinates	Search for a visible element in the page which matches the coordinates
POST	/session/$sessionId/touch/clickByCoordinate	Clicks on the visible element which matches the coordinates
POST	/quamotion/app	Adds an app to the Quamotion WebDriver app repository.
GET	/quamotion/app	Gets all applications in the Quamotion WebDriver app repository
GET	/quamotion/device	Gets all devices which are available to test
POST	/quamotion/device/$deviceId/app/$appId	Installs the application on the device
POST	/quamotion/device/$deviceId/app/$appId/$appVersion	Installs the application on the device
GET	/quamotion/device/$deviceId/app	Gets all installed applications on the device
DELETE	/quamotion/device/$deviceId/app/$appId	Uninstall the application from the device

DELETE	/quamotion/device/$deviceId/app/$appId/$appVersion	Uninstall the application from the device
DELETE	/quamotion/device/$deviceId/app/$appId/settings	Deletes the settings of the application installed on the mobile device
DELETE	/quamotion/device/$deviceId/reboot	Reboots the device

Scripts can be written in multiple languages. Currently there are Powershell, Java and C# clients available.

In Powershell a script looks as follows:

```
Click-Element -xpath "input[@name='email']"
Enter-Text "bart.saintgermain@quamotion.mobi"
Click-Element -xpath "input[@name='pass']"
Enter-Text "a strong password"
Click-Element -xpath "button[@name='login']"
```

The script above automates the login activity. It first clicks on the email text field and enters the email. Next it clicks on the password field and enters the password. Finally, a click is performed on the login button.

As can be seen from the sample script, XPaths are used to identify elements in the screen. The Quamotion WebDriver provides two tools to find the XPath for a specific screen element or to create the script by recording (see section 7.5.1 and 7.5.2).

TAP support

To add support for app interaction automation in a TAP test plan, a TAP plugin has been developed. This plugin includes a TAP instrument that represents the Quamotion WebDriver server, and a set of test steps to interact with the UE through this server.

Table 3 shows the description and settings for the QuamotionWebDriverInstrument. This TAP instrument is in charge of keeping track of the sessions opened with the WebDriver. A session represents an app running on a device, and it is necessary to have an open session in order to send commands that interact with the app. The current implementation only allows one session to be active at a time.

Table 3 Quamotion WebDriver TAP plugin instruments

Instrument	**Setting**	**Description**
Quamotion WebDriver (QuamotionWebdriver-Instrument)		Handles connection to a Quamotion Webdriver server. Creates and tracks sessions, i.e. an app running on a device.
	WebDriver / Base URL	URL prefix where the WebDriver API calls can be made to. This would typically be http://localhost:17894/wd/hub/ for a Quamotion WebDriver server running in the same host as TAP.
	Sessions / Session ready: poll period	When opening a new session, its status has to be polled periodically. This setting controls the polling period.
(ICsvInstrument)	Sessions / Session ready: max retries	When polling the status of a session that is being opened, poll until the session is ready or the maximum number of retries is reached.

Table 4 shows a subset of the test steps that are provided by the TAP plugin. These steps can be divided into three categories:

- Session management: open and close sessions. It is mandatory to have an open session before performing any further actions.
- User actions: simulate the interaction of a user with the app, e.g. tapping on an element, or entering text on a text field.
- Queries: query the current state of the screen, e.g. query if an element is present.

Table 4 Quamotion WebDriver TAP plugin test steps

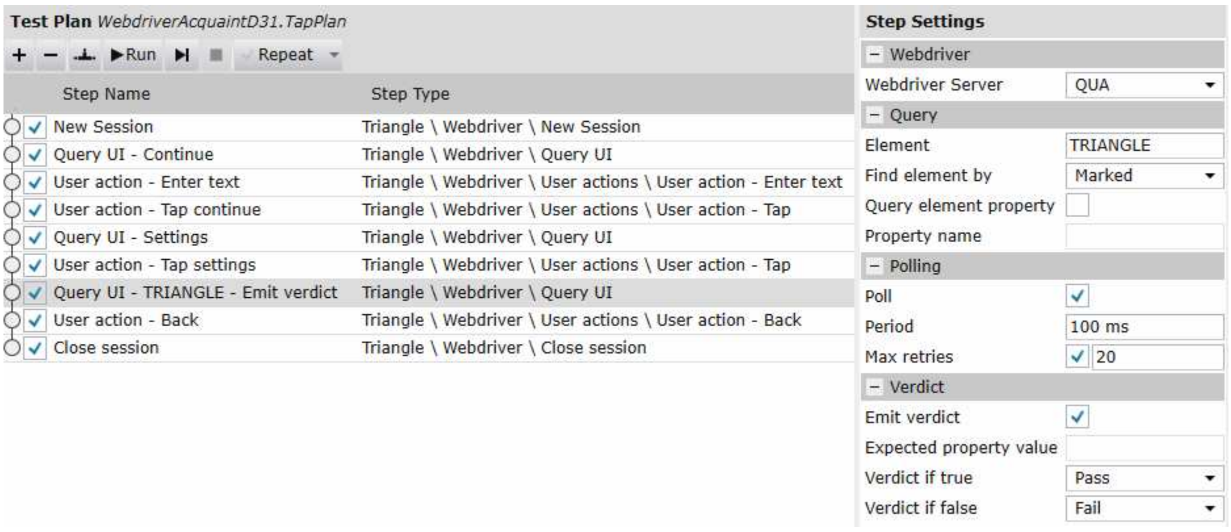
Test step	**Setting**	**Description**
New session (NewSessionStep)		Opens a new session in the WebDriver.
	WebDriver / WebDriver Server	QuamotionWebDriverInstrument on which the session will be opened.
	Session / Device ID	ID of the device where the app will be executed and automated. This ID can be obtained on the Quamotion WebDriver web interface, or using the adb command line tool.
	Session / Device ID	ID of the app that will be executed and automated on the device. This ID can be obtained on the Quamotion WebDriver web interface.
	Session / Clear app settings	Clean the app settings before starting the session. This ensures that the actions performed next will not be influenced by past tests.
	Advanced / Close all sessions	Close all active sessions before opening a new one.
	Advanced / Reuse session	Reuse an existing open session, if possible. This accelerates the start-up process. On the other hand, the app will not be restarted, and therefore may be in an unknown state.
Close session (CloseSessionStep)		Closes the currently active session in the given WebDriver.
	WebDriver / WebDriver Server	The QuamotionWebDriverInstrument on which the currently active session will be closed.
User action – Tap (TapStep)		Taps on an element in the screen. This can also be referred as clicking.
	WebDriver / WebDriver Server	QuamotionWebDriverInstrument which will perform the action on the app.
	Action / Element	An identifier for the element. The identifier can be any of the supported by the different “find strategies” allows by WebDriver.
	Action / Find element by	Strategy that will be used to find the element with which to interact. Depending on this strategy, the element identifier will be interpreted differently.
	Wait / Wait for	If disabled, try to perform the action immediately.

	element	If enabled, the action will be delayed until the element is found on the screen.
	Wait / Polling period	Polling period when checking the presence of the element.
	Wait / Max retries	Maximum number of polling retries when checking the presence of the element.
User action – Long press (LongPressStep)		Tap and hold an element in the screen for a small period of time, also known as long pressing. This step shares the same settings as TapStep.
User action – Enter text (EnterTestStep)		Enter a text, i.e. a sequence of keyboard presses, in an element in the screen, e.g. a text field. This step has all the settings as TapStep, plus the following ones.
	Action / Text	Text that will be entered in the element.
User action – Back (BackStep)		Goes back in the app. For instance, in Android this means pressing the "back" button.
	WebDriver / WebDriver Server	QuamotionWebDriverInstrument which will perform the action on the app.
Query UI (QueryUiStep)		Queries the app UI to find a specific element, e.g. a button or text. Additionally, it can query a public property of that element. A result will be produced and, optionally, a verdict.
	WebDriver / WebDriver Server	QuamotionWebDriverInstrument which will query the app UI.
	Query / Element	An identifier for the element. The identifier can be any of the supported by the different "find strategies" allows by WebDriver.
	Query / Find element by	Strategy that will be used to find the element with which to interact. Depending on this strategy, the element identifier will be interpreted differently.
	Query / Query element property	If enabled, query the following public property of the element, if found.
	Query / Property name	Name of a public property from the selected element.
	Polling / Wait for element	If disabled, check the presence of the element just once. If enabled, the presence of the element will be checked periodically.
	Polling / Period	Polling period when checking the presence of the element.
	Polling / Max retries	Maximum number of polling retries when checking the presence of the element.
	Verdict / Emit verdict	If enabled, the step will emit one of the verdicts configured next. If the element is present (or the expected value matches the actual value of the public property), then "Verdict if true" will be emitted. Otherwise, "Verdict if

		false" will be emitted.
	Verdict / Expected property value	If querying a public property of the element, compare its value against this expected value, as strings.
	Verdict / Verdict if true	The verdict to emit if the UI element is present (or the property value matches the expected value).
	Verdict / Verdict if false	The verdict to emit if the UI element is not present (or the property value does not match the expected value).

The TAP plugin has been implemented using the Quamotion Webdriver Client library for C# [8]. This library extends the existing Selenium WebDriver library with additional APIs to support the custom features of the Quamotion WebDriver.

Figure 15 shows an example of the usage of these test steps to automate the Acquaint app from Xamarin. In this test plan, several user actions are performed to set up the app, and later to check the value of one of the text fields in the settings screen. If the value is found, then the test case passes. Otherwise, the test case is considered to have failed.



Step Name	Step Type
✓ New Session	Triangle \ Webdriver \ New Session
✓ Query UI - Continue	Triangle \ Webdriver \ Query UI
✓ User action - Enter text	Triangle \ Webdriver \ User actions \ User action - Enter text
✓ User action - Tap continue	Triangle \ Webdriver \ User actions \ User action - Tap
✓ Query UI - Settings	Triangle \ Webdriver \ Query UI
✓ User action - Tap settings	Triangle \ Webdriver \ User actions \ User action - Tap
✓ Query UI - TRIANGLE - Emit verdict	Triangle \ Webdriver \ Query UI
✓ User action - Back	Triangle \ Webdriver \ User actions \ User action - Back
✓ Close session	Triangle \ Webdriver \ Close session

− Webdriver	
Webdriver Server	QUA
− Query	
Element	TRIANGLE
Find element by	Marked
Query element property	
Property name	
− Polling	
Poll	✓
Period	100 ms
Max retries	✓ 20
− Verdict	
Emit verdict	✓
Expected property value	
Verdict if true	Pass
Verdict if false	Fail

Figure 15 Example of TAP using WebDriver TAP plugin

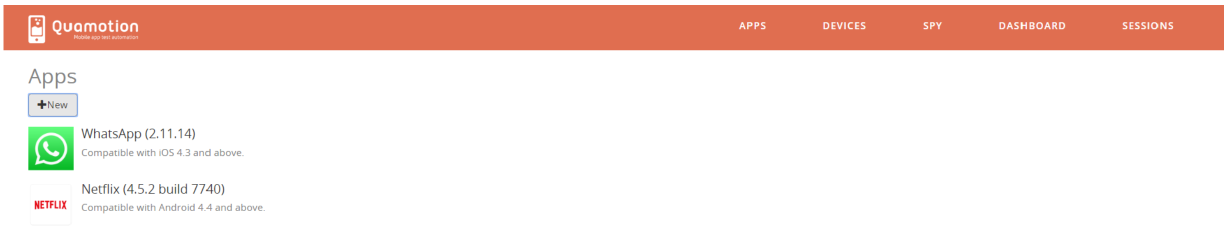
4.2.2 Remote device control

On top of the Quamotion WebDriver a Mobile Application Frontend facilitates the creation of user actions test scripts. This frontend is fully web based.

There are three main components: app management (section 7.4.1), device management (section 7.4.2), and app inspection (section 7.5).

App management

With the App management you can list, add and remove applications to the Quamotion WebDriver internal application store. In order to be installed and used on a device, apps have to be added here first. Information about the application e.g. unique application id, version number, can be retrieved.



Apps

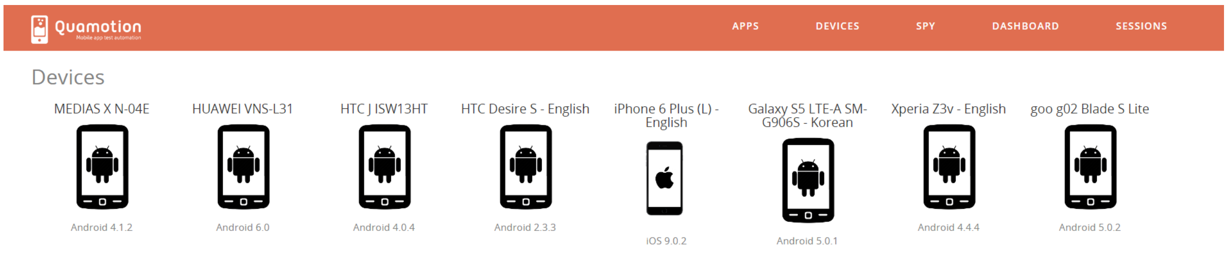
+New

WhatsApp (2.11.14)
Compatible with iOS 4.3 and above.

Netflix (4.5.2 build 7740)
Compatible with Android 4.4 and above.

Device management

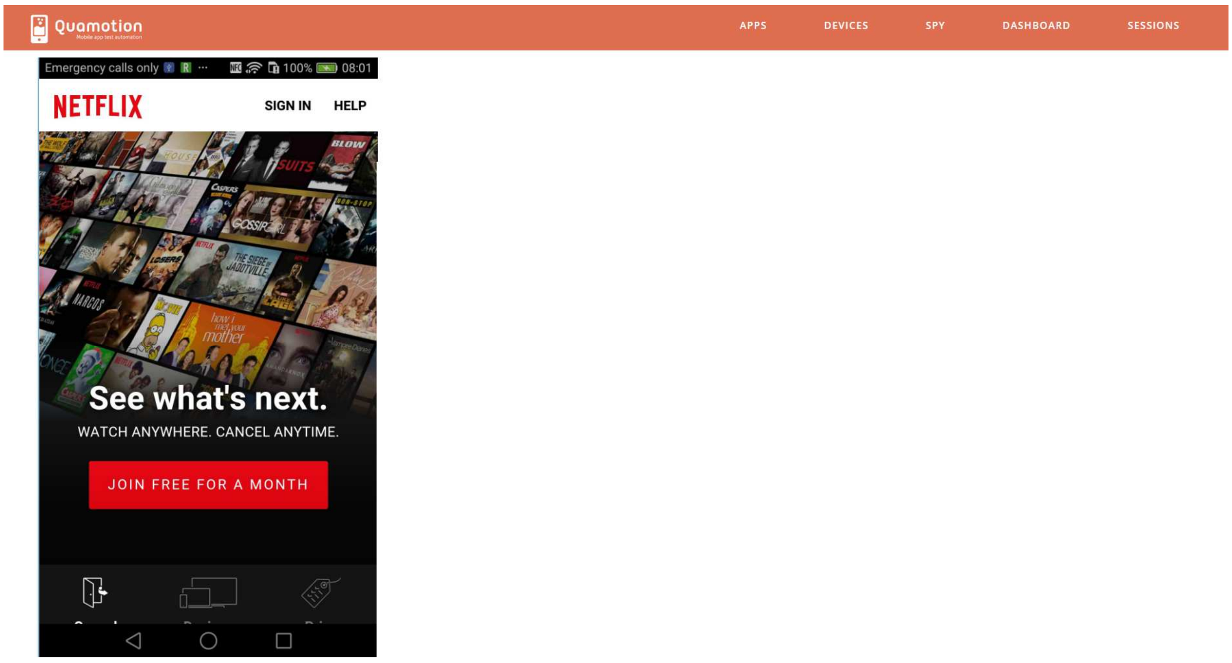
The device management lists all devices connected to the computer running the Quamotion WebDriver. Basic information about the device can be retrieved, e.g., the serial number of the device.



Devices

MEDIAS X N-04E	HUAWEI VNS-L31	HTC J ISW13HT	HTC Desire S - English	iPhone 6 Plus (L) - English	Galaxy S5 LTE-A SM-G906S - Korean	Xperia Z3v - English	goo g02 Blade S Lite
Android 4.1.2	Android 6.0	Android 4.0.4	Android 2.3.3	iOS 9.0.2	Android 5.0.1	Android 4.4.4	Android 5.0.2

Selecting the device gives you the ability to launch a remote device screen, which enables to view the device screen remotely. Controlling the device remotely will be added later. Common gestures will be made available to mimic the behaviour of a real user interacting with the physical device, e.g. tapping on the remote screen triggers a tap on the remote device.



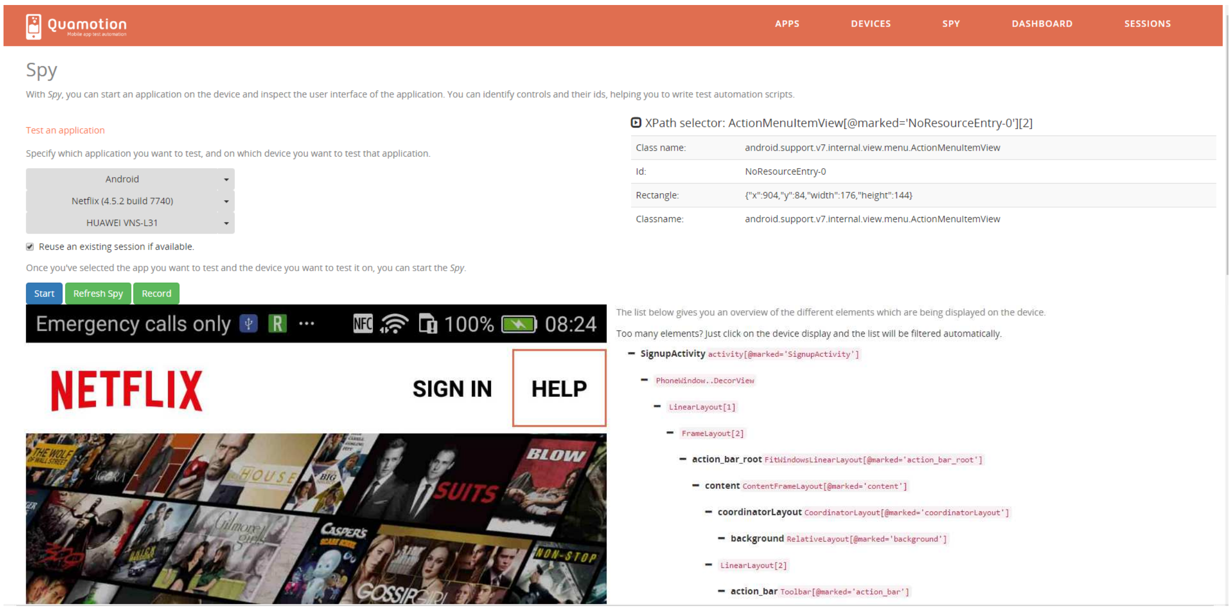
4.2.3 User interaction automation tools

The Quamotion WebDriver frontend exposes two application inspection tools. One is the Spy and the second is the Recorder.

Spy

Based on an active session and the remote device display, users are able to click on an element in the screen and retrieve all information about the user interface widget being selected. The spy shows the following information

- The XPath of the widget identifying uniquely the widget, which can then be used in the script to find elements
- A tree showing the ancestors of the widget.
- All available properties of the selected widget
- The widget is highlighted in the remote device screen



Spy

With *Spy*, you can start an application on the device and inspect the user interface of the application. You can identify controls and their ids, helping you to write test automation scripts.

Test an application

Specify which application you want to test, and on which device you want to test that application.

Android
Netflix (4.5.2 build 7740)
HUAWEI VNS-L31

☑ Reuse an existing session if available.

Once you've selected the app you want to test and the device you want to test it on, you can start the *Spy*.

XPath selector: ActionMenuItemView[@marked='NoResourceEntry-0'][2]

Class name:	android.support.v7.internal.view.menu.ActionMenuItemView
Id:	NoResourceEntry-0
Rectangle:	{"x":904,"y":84,"width":176,"height":144}
Classname:	android.support.v7.internal.view.menu.ActionMenuItemView

The list below gives you an overview of the different elements which are being displayed on the device.

Too many elements? Just click on the device display and the list will be filtered automatically.

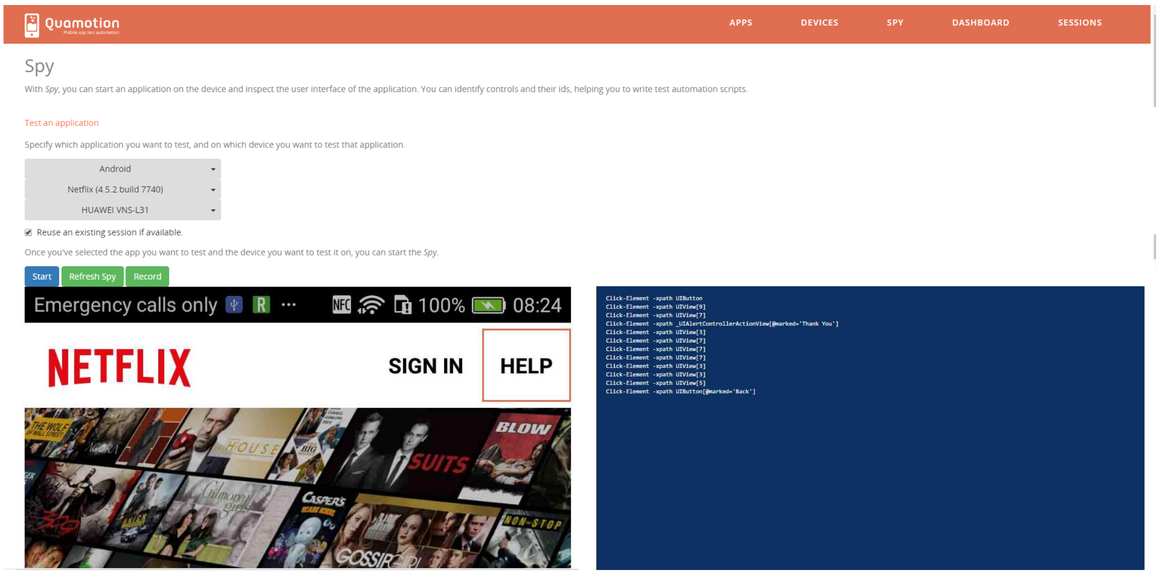
```
- SignupActivity activity[@marked='SignupActivity']
  - PhoneWindow..DecorView
    - LinearLayout[1]
      - FrameLayout[2]
        - action_bar_root FitWindowsLinearLayout[@marked='action_bar_root']
          - content ContentFrameLayout[@marked='content']
            - coordinatorLayout CoordinatorLayout[@marked='coordinatorLayout']
              - background RelativeLayout[@marked='background']
            - LinearLayout[2]
              - action_bar Toolbar[@marked='action_bar']
```

Recorder

The Quamotion frontend provides currently a basic version on the Recorder.

The recorder generates a script based on the user interactions on the device. Gestures like Tapping and entering text are recorded. For each user action on the device a statement is generated (currently in raw JSON format or Powershell statements).

Note that the recorder will not generate automatic checkpoints. After recording, the script needs to be fine-tuned and enhanced manually



Spy

With *Spy*, you can start an application on the device and inspect the user interface of the application. You can identify controls and their ids, helping you to write test automation scripts.

Test an application

Specify which application you want to test, and on which device you want to test that application.

Android
Netflix (4.5.2 build 7740)
HUAWEI VNS-L31

☑ Reuse an existing session if available.

Once you've selected the app you want to test and the device you want to test it on, you can start the *Spy*.

Start | Refresh Spy | Record

```
Click-Element -xpath UIButton
Click-Element -xpath UIView[9]
Click-Element -xpath UIView[7]
Click-Element -xpath _UIAlertControllerActionView[@marked='Thank You']
Click-Element -xpath UIView[3]
Click-Element -xpath UIView[7]
Click-Element -xpath UIView[7]
Click-Element -xpath UIView[7]
Click-Element -xpath UIView[3]
Click-Element -xpath UIView[3]
Click-Element -xpath UIView[5]
Click-Element -xpath UIButton[@marked='Back']
```

4.3 Adb

Android Debug Bridge (adb) [25] is a command-line tool provided by the Android SDK. It lets the user communicate with an Android device to perform actions on it, or to send commands to an app. The tool follows a client-server approach, with clients that send commands, daemons that runs the commands on a device, and a server that handles the communication between clients and daemons. The daemons run on the Android devices, will the clients and server typically run on a development machine.

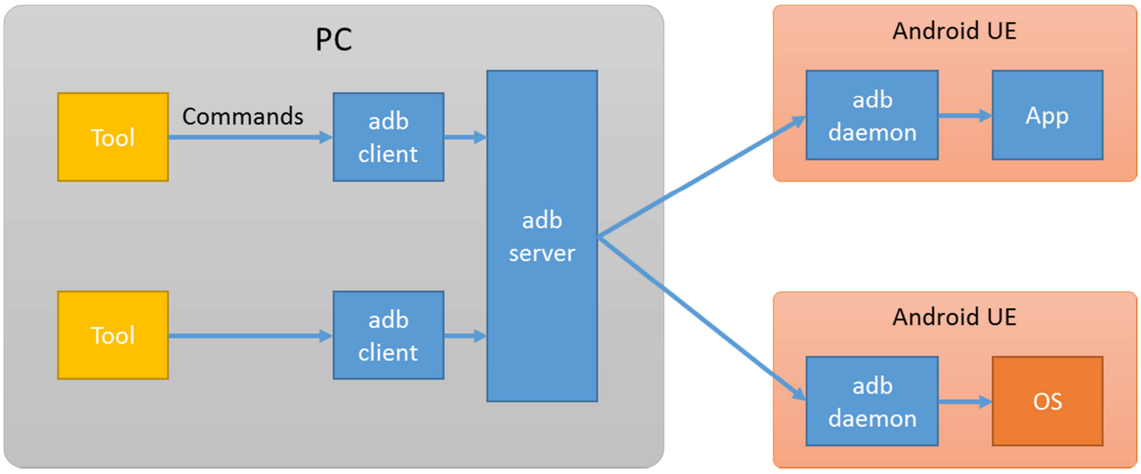


Figure 16 Adb usage scenario

Many tools use adb to interact with an Android device or the apps running on it. For instance, Quamotion WebDriver uses adb to send the user actions as commands that can be performed on the device. TestelDroid (introduced in Section 5) is controlled through intents (inter-app messages used in Android) sent using adb.

4.4 PTP synchronization

Precise Time Protocol (PTP), also known as IEEE 1588, is a communication standard used for clock synchronization between different instruments/machines. The expected accuracy of PTP in a local area is in the order of microseconds. PTP is designed for systems with stricter requirements than what Network Time Protocol (NTP) protocol can provide.

PTP will used in the testbed to keep accurate time synchronization between the different instruments in the testbed.

5 Measurements and data collection

The testbed has integrated two tools, TestelDroid and AT4 Performance Test Tool, which enable the instrumentation of mobile devices to collect data traffic information and OS API information. The testbed also includes a N6705B power analyzer to collect energy measurements from UEs. To collect measurements and information from apps which cannot be accessed in any other way, an instrumentation library will be provided to app developers. Finally, all the collected measurements will be sent to a central OML server.

5.1 TestelDroid monitoring tool

5.1.1 Introduction

The objective of TestelDroid is to take advantage of the engineering features provided by current commercial smartphones for the development of advanced monitoring tools for mobile devices.

5.1.2 Functionalities

Current features offered by the tool are the following:

- Network information: Current operator, RAT (Radio Access Technology), cell identity, LAC (Location Area Code), RSSI (Radio Signal Strength Indicator), PSC (Primary Scrambling Code).
- Neighboring cell information: PSC, RSSI and type of network (not available for Samsung based phones, such as Samsung Galaxy S or Nexus S).
- GPS information: Longitude, latitude, altitude and speed.
- Traffic: Network traffic (monitoring mode displays only some information of the packet, such as protocol, IP source/destination or ports involved), using tcpdump. TestelDroid provides pcap format files containing the traffic captured.

  Besides monitoring and logging, TestelDroid allows:
- Connectivity test: In order to diagnose connectivity issues
  - o Ping a host (ping options are configurable)
  - o Test if a port is open on a specified host
- Traffic test: Server-Client model, allows the transfer of an auto-generated file (size can be specified) between two devices. Speed is monitored on the server side and an average size is provided upon completion of the file transfer

5.1.3 Control and configuration API

To enhance its integration in the testbed, a number of extensions have been recently added to TestelDroid. The extensions include support for Standard Commands for Programmable Instruments (SCPI), cOntrol and Management Framework (OMF) and OMF Measurement Library (OML). SCPI is the most widespread interface for measurement equipment control in many areas. OMF and OML extensions enable powerful orchestration framework languages that reduce the time required to define tests.

Table 5 TestelDroid configuration and control API

Command	**Purpose**	**Q**
**RST*	Reset the System	N
**IDN?*	Return a string with the name and version of	Y

	the application	
**STB?*	Return a byte with the status, e.g.,: 0x00: idle, 0x01: running, 0xFF: error	Y
SETup:NETwork:RESTart	Restart the network (e.g.: using flight mode, searching network, etc.)	N
SETup:MEASurement:START	Start a measurement session (which includes starting the service under test)	Y
SETup:MEASurement:STOP	Stop a measurement session (which includes stopping the service under test)	Y
SETup:MEASurement:CONF: NETwork:ENable	Enable/Disable the NETWORK measurement (default on)	Y
SETup:MEASurement:CONF:TRAFfic: ENable	Enable/Disable the TRAFFIC measurement (default on)	Y
SETup:MEASurement:CONF:GPS:ENable	Enable/Disable the GPS reading (default off)	Y
SETup:MEASurement:CONF: NEIGHbour:ENable	Enable/Disable the NEIGHBOUR measurement (default off)	Y
SETup:MEASurement:CONF:PROFile:ENable	Enable/Disable the PROFILE measurement (default on)	Y
SETup:MEASurement:CONF:PROFile: SCENario	Add information about the context in which measurements are collected: vehicular, static, pedestrian or high-speed. This information is provided by the user of the tool. This information is very useful during the analysis and interpretation of the information collected by TestelDroid.	Y
SETup:MEASurement:CONF:PROFile:TECH	Define the network access technology, can be Mobile (generic and default), GSM, HSPA, LTE, UMTS,or Wi-Fi	Y
SETup:MEASurement:CONF:PROFile: CONFiguration	Add information about the context in which measurements are collected: Fixed-Fixed, Mobile-Mobile, Mobile-Fixed. This information is provided by the user of the tool. This information is very useful during the analysis and interpretation of the information collected by TestelDroid	Y
SETup:MEASurement:CONF:PROFile:SUBID	String identficating the SubId that will be used to generate the capture id, which has the following format YYYYMMDDHHSUBID	Y
SETup:MEASurement:CONF:PROFile:PEERID	String identifying the PeerId of the node, which can be 1 or 2, or NA to disable the use of PeerIds	Y
SETup:MEASurement:CONF:PROFile: COMMENTS	To add comments from the user. This information is provided by the user of the tool. This information is very useful during the analysis and interpretation of the information collected by TestelDroid	Y
RETrieve:MEAS:NETwork?	Returns the network file, the format might be:	Y

	INTEGER,ASCII_STRING,INTEGER,BINARY_STRING First integer is the length of the filename, the ascii_string is the filename, the next integer is the file size and the binary string is the content of the file. Network file format: <timestamp> <RAT> <Cell ID> <LAC> <RSSI> <MCC+MNC> <PSC>	
RETrieve:MEAS:GPS?	Returns the GPS file in the same format than the NETwork file. GPS file format: <timestamp> <latitude> <longitude> <speed>	Y
RETrieve:MEAS:BATTERY?	Returns the BATTERY file in the same format than the NETwork file. Battery file format: <timestamp> < µW > (available only when kernel has been modified)	Y
RETrieve:MEAS:NEIGHbour?	Returns the NEIGHBOUR file in the same format than the NETwork file. Neighbour file format: <timestamp> <PSC> <RSSI> <RAT>	Y
RETrieve:MEAS:PROFile?	Returns the PROFILE file in the same format than the NETwork file. Profile file format: <label>context_comment</label>	Y
RETrieve:MEAS:TRAFFIC?	Returns the TRAFFIC file in the same format that the NETWORK file Traffic forma file: pcap	Y

5.2 AT4 Performance Tool

5G test scenarios will require high resolution for reporting target QoS KPIs. The Triangle Testing Framework will provide up to layer 7 SDU packet resolution in the computation of data performance KPI thanks to the integration of the AT4 Performance Tool.

This tool is composed of two components, Controller and Agents (data endpoints), and uses proprietary mechanisms to synchronize the Agents and provides accurate one-way measurements.

This tool includes a built-in traffic generator with the capability of generating constant rates, ramps, loops and statistical traffic patterns which is something of utmost importance for setting up the desired environment in terms of varying traffic loads (e.g., for measuring LTE-U impact on Wi-Fi networks).

Additionally, this tool has the ability to automate some mobile Apps on Android devices and measuring relevant QoE KPI such as YouTube buffering occurrences.

The summary of measurement capabilities provided to the Triangle Testing Framework in Release 1 is summarized in Appendix 1.

This Tool is currently integrated in the framework by a CLI based interface which is specified in Appendix 2.

This interface has limited functionality and formal Remote Control Interface shall be implemented within the context of Triangle project to be used in further releases of the testing framework.

The AT4 wireless Performance Tool provides to the first release of the Triangle Testing Framework: the QoS and QoE measurements which are described in the following sections.

5.2.1 One-way Delay

This packet level measurement is an implementation of the RFC 2679. This measurement is written in C for performance and portability across the range of supported mobile device platforms.

The measurement methodology proceeds as follows:

1. The system arranges that Source AT4-Agent and Destination AT4-Agent hosts are synchronized; that is, that they have clocks that are very closely synchronized with each other and each fairly close to the actual time.
2. At the Source AT4-Agent, the system selects Source and Destination IP addresses, and forms a test packet with these addresses. A test packet uses UDP transport protocol. The content of the test packet is random. The size of the packet (SDU) is parameter of the system and the packets inter departure time (i.e., time between consecutive packets) are parameters of the system.
3. At the Destination AT4-Agent, the system arranges to receive the packet.
4. At the Source AT4-Agent host, the system places a timestamp (t_1) in the prepared packet, and sends it towards the Destination AT4-Agent host.
5. If the packet arrives within a reasonable period of time, the system takes a timestamp (t_2) as soon as possible upon the receipt of the packet. Note that the threshold of 'reasonable' is a parameter of the system.
6. Figure 17 shows in the protocol stack of the AT4-Agent hosts where the system measures the timestamps t_1 and t_2.

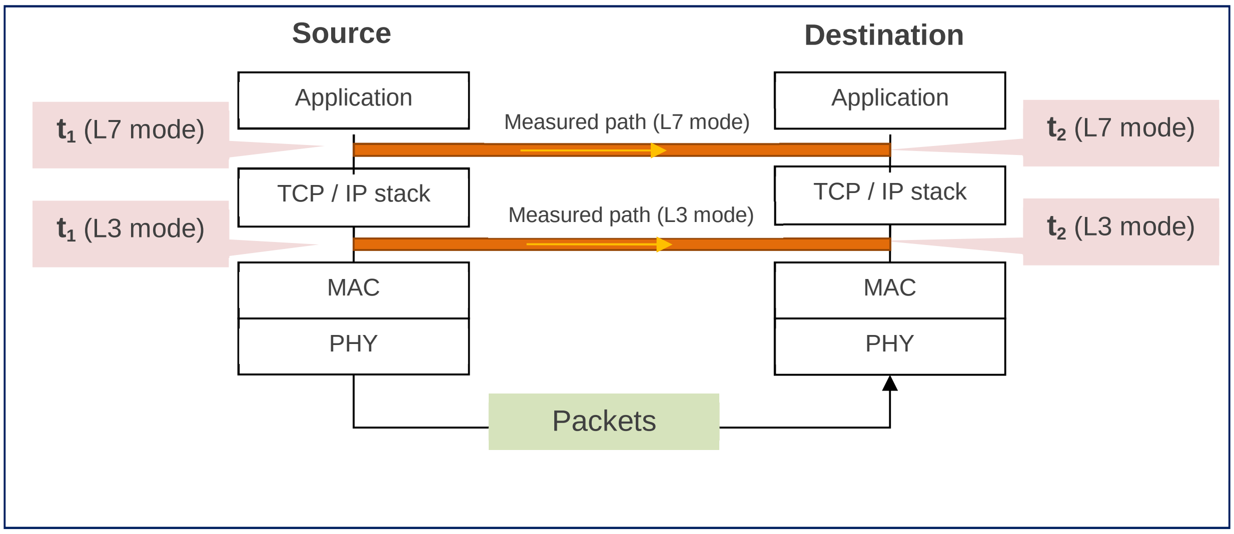


Figure 17 Timestamps for OWD measurements

7. By subtracting the two timestamps, an estimate of one-way delay can be computed for a give packet 'i':

$$\mathrm{OWDi} = t_2 - t_1 \qquad (1)$$

8. If the packet fails to arrive within a reasonable period of time, the one-way delay is taken to be undefined (informally, infinite).
9. The system groups the OWD_i samples into fixed periods of time (dT). The period “dT” is a parameter of the system. Default value is 1 second.
10. Given OWD [T, dT] = {OWD_1, OWD_2, …, OWD_N} the set of OWD_i samples within the partial interval [T, dT], the system computes the following instantaneous (i.e., per interval) statistics:

$$\text{Average OWD } [T, dT] = \frac{1}{N}\sum_{i=dT}^{T+dT} OWD_i \qquad (2)$$

$$\text{Minimum OWD } [T, dT] = \min \text{ of } \{OWD\ [T, dT]\} \qquad (3)$$

$$\text{Xth Percentile OWD } [T, dT] = PCTL_X\{OWD\ [T, dT]\} \qquad (4)$$

11. Given OWD [t_{start}, t_{end}] = {OWD_1, OWD_2, … OWD_M} the set of OWD_i samples within the total measurement interval (t_{start}, t_{end}), the system computes the following statistics:

$$\text{Xth Percentile OWD } = PCTL_X\{OWD\ [t_{start}, t_{end}]\} \quad (5)$$

$$\text{PDF OWD } = \text{HISTOGRAM } \{OWD\ [t_{start}, t_{end}]\} \qquad (6)^1$$

$$\text{Average OWD } = \frac{1}{M}\sum_{i=tstart}^{tend} OWD_i \qquad (7)$$

Table 6 summarizes the One-way Delay KPIs reported by the Triangle Testing Framework release 1.

Table 6 One-way Delay KPIs

KPI	**X-Axis**		**Y-Axis**	**Data Source**
OWD Average (t)	time	dT (ms)	Average OWD [T, dT]	(2)
OWD Minimum (t)	time	dT (ms)	Minimum OWD [T, dT].	(3)
OWD Median (t)	time	dT (ms)	50th Percentile OWD [d, dT]	(4)
OWD Percentile (t)	time	dT (ms)	0th Percentile OWD [d, dT] 5th Percentile OWD [d, dT] … 100 th Percentile OWD [d, T]	(4)
OWD CDF	%	*1%*	Xth Percentile OWD	(5)

[1] One OWD PDF plot is calculated for the OWD samples within the total measurement. For example, given a test 60 s long, where 10000 packets are correctly received, then the system obtains the following vector: OWD [0, 60] = {OWD_1, …., OWD_{10000}} and one PDF histogram is calculated for that vector of 10000 samples.

OWD PDF	frequency	Histogram OWD	(6)

5.2.2 One-way Delay Variation (Jitter)

This packet level measurement is an implementation of the RFC 3393. This measurement is written in C for performance and portability across the range of supported platforms.

The methodology proceeds as follows:

1. Repeat steps 1 to 7 from the One-way delay measurement methodology to obtain the vector of samples OWD.
2. The system computes the IPDV samples as follows:

$$\mathrm{IPDV_i = OWD_i - \ OWD_{i-1}} \qquad (1)$$

IPDV$_i$ is undefined if either OWD$_i$ or OWD$_{i-1}$ is undefined (packet loss).

3. The system groups the IPDV$_i$ samples into fixed periods of time (dT). The period “dT” is a parameter of the system. Default value is 1 second.
4. Given IPDV [T, dT] = {IPDV$_1$, IPDV$_2$, …, IPDV$_N$} the set of IPDV$_i$ samples within the partial interval [T, dT], the system computes the following instantaneous (i.e., per interval) statistics:

$$\text{Average IPDV }[\mathrm{T, dT}] = \frac{1}{\mathrm{N}}\sum_{\mathrm{i=dT}}^{\mathrm{T+dT}} \mathrm{IPDV_i} \qquad (2)$$

$$\text{Peak to Peak IPDV }[\mathrm{T, dT}] = \text{max of }\{\mathrm{OWD}\ [\mathrm{T, dT}]\} - \ \text{min of }\{\mathrm{OWD}\ [\mathrm{T, dT}]\} \ (3)$$

$$\text{Xth Percentile IPDV }[\mathrm{T, dT}] = \mathrm{PCTL_X}\{\mathrm{IPDV}\ [\mathrm{T, dT}]\} \qquad (4)$$

5. Given IPDV [t$_{start}$, t$_{end}$] = {IPDV$_1$, IPDV$_2$, … IPDV$_M$} the set of IPDV$_i$ samples within the total measurement interval (t$_{start}$, t$_{end}$), the system computes the following statistics:

$$\text{Xth Percentile IPDV } = \mathrm{PCTL_X}\{\mathrm{IPDV}\ [\mathrm{t_{start}, t_{end}}]\} \qquad (5)$$

$$\text{Average IPDV } = \frac{1}{\mathrm{M}}\sum_{\mathrm{i=tstart}}^{\mathrm{tend}} \mathrm{IPDV_i} \qquad (6)$$

Table 7 summarizes the One-way Delay Variation KPIs reported by the Triangle Testing Framework release 1.

Table 7 One-way Delay Variation KPIs

KPI	**X-Axis**		**Y-Axis**	**Data Source**
IPDV Average (t)	time	dT (ms)	Average IPDV [T, dT]	(2)
IPDV Peak to Peak (t)	time	dT (ms)	Peak to Peak IPDV [T, dT].	(3)
IPDV Percentile (t)	time	dT (ms)	0th Percentile IPDV [d, dT] 5th Percentile IPDV [d, dT] … 100 th Percentile IPDV [d, T]	(4)

IPDV CDF	%	*5%*	Xth Percentile IPDV	(4)
IPDV PDF	*frequency*		Histogram IPDV	(5)

5.2.3 One-way Packet Loss Rate

This packet level measurement is an implementation of the RFC 2680. This measurement is written in C for performance and portability across the range of supported platforms.

The methodology proceeds as follows:

1. Repeat steps 1 to 4 from the One-way delay measurement.
2. If a packet 'i' arrives within a reasonable period of time, the system counts $PL_i = 0$. Otherwise, if a packet 'i' does not arrive within a reasonable period of time, the system counts $PL_i = 1$. Note that the threshold of 'reasonable' is a parameter of the system.
3. The system groups the PL_i samples into fixed periods of time (dT). The period "dT" is a parameter of the system. Default value is 1 second.
4. Given PL [T, dT] = {PL_1, PL_2, …, PL_N} the set of PL_i samples within the partial interval [T, dT], the system computes the following instantaneous (i.e., per interval) statistics:

$$\text{PL Rate}\,[\text{T}, \text{dT}] = \frac{1}{\text{N}} \sum_{\text{i}=\text{dT}}^{\text{T}+\text{dT}} \text{PL}_\text{i} \times 100 \qquad (1)$$

5. Given PL [t_{start}, t_{end}] = {PL_1, PL_2, … PL_M} the set of PL_i samples within the total measurement interval (t_{start}, t_{end}), the system computes the following statistics:

$$\text{PL Rate} = \frac{1}{\text{M}} \sum_{\text{i}=\text{tstart}}^{\text{tend}} \text{PL}_\text{i} \times 100 \qquad (2)$$

Table 8 summarizes the One-way Packet Loss Rate KPIs reported by the Triangle Testing Framework release 1.

Table 8 One-way Packet Loss Rate KPIs

KPI	**X-Axis**		**Y-Axis**	**Data Source**
PL Rate (t)	time	dT (ms)	Average PL [T, dT]	(1)

5.2.4 One-way Packet Loss Distribution

This packet level measurement is an implementation of the RFC 3357. This measurement is written in C for performance and portability across the range of supported platforms.

The methodology proceeds as follows:

1. Repeat steps 1 to 3 from the One-way packet loss measurement to obtain the PL vector.
2. Given *PL* [t_{start}, t_{end}] = {PL_1, PL_2, … PL_M} the set of PL_i samples within the total measurement interval (t_{start}, t_{end}), the system computes the following:

a. Loss Distance LD_i: When a packet is considered lost (PL_j= 1), the system looks at its sequence and compare it with that of the previous lost packet (PL_k =1). The difference j-k is the *loss distance* between the lost packet and the previous lost packet,

b. Loss Period LP_i: A loss period begins if $PL_j = 1$ and $PL_{j-1} = 0$.

c. Noticeable Loss: A packet loss is "noticeable" if LD_i is no greater than delta, a positive integer, where delta is the "loss constraint".

3. Given PL, LD and LP vectors, the system compute the following:

$$\text{Noticeable Rate (\%)} = \frac{\text{Number of noticeable losses}}{\text{Total number of losses}} \text{ for a given "delta"} \qquad (1)$$

$$\text{Loss Period Lengths} = \text{Number of packet lost in each loss period} \qquad (2)$$

$$\text{Inter} - \text{loss Period Lengths} = \text{Distance between consecutive periods loss} \qquad (3)$$

$$\text{Period Loss Duration} = \text{Duration of each loss period} \qquad (4)$$

Table 9 summarizes the One-way Packet Loss Distribution KPIs reported by the Triangle Testing Framework release 1.

Table 9 One-way Packet Loss Distribution KPIs

KPI	**X-Axis**		**Y-Axis**	**Data Source**
One-way Loss Noticeable Rate	Delta		Noticeable Rate (%)	(1)
One-way Consecutive Lost Packets	Length		Periods Loss Frequency	(2)
One-way Consecutive Lost Packets Distance	Length		Inter Periods Loss Frequency	(3)
One-way Consecutive Lost Packets Time (s)	time	dT (ms)	Period Loss duration (ms)	(4)

5.2.5 YouTube™

The AT4-Agent (section 5.2) forces the mobile device to visualize a video from YouTube™ and measures the quality of experience of the video streaming.

The implementation of this measurement is driven by the official YouTube API which embeds a player on the AT4-Agent that most closely behaves like the YouTube native application[2].

A YouTube session consists of a number of individual YouTube playbacks.

1. AT4-Agent loads a full screen Web View component.

[2] At the time this document has been revised.

2. AT4-Agent gets a locally stored HTML test page. This HTML page contains Javascript code using YouTube iFrame API. It implements all required features for the measurement:
    a. It loads a YouTube player.
    b. It sets the size of the player to the size of the testing device screen which is equivalent to a landscape full screen playback.
    c. It plays the video clip (e.g., auto play).
    d. It captures the user events required to calculate the KPIs (quality changes and player states).
    e. AT4-Agent plays the video in the embedded player in the loaded web view.

Table 10 summarizes the KPIs reported by the Triangle Testing Framework release 1 for the reference App YouTube^{TM}.

Table 10 YouTube^{TM} KPIs

KPI	**Units**	**Description**
MOS	-	Estimation of the video quality as perceived by a user. Possible values: 1 (bad) to 5 (good). This estimation is based on the initial buffering, and the re-buffering index.
Playback Time	s	This is the actual duration of the playback from loading the player until the video clip ends. Playback Duration = Initial Buffering Time + Total Re-buffering Time + Other impairments Other impairments: - Video lagging due to testing device CPU over load. - iFrame API accuracy in rebufferings: Elapsed time from playback pauses until API event calls `onPlayerStateChange` == `PlayerState.BUFFERING`
Playback Size	MB	The amount of data downloaded by the device to play the video regardless of the bit rate used. Therefore, the same video clip may report different video sizes in different test iterations if the bit rate used has been also different.
Initial buffering	s	The period between the starting time of loading a video and the starting time of playing it.
Re-bufferings	-	When the buffered video data decreases to a low value, the playback will pause, and the player will enter into a re-buffering state. This KPI shows how many times this event happens. The following KPIs are derived from all the re-bufferings in a video playback: - Maximum re-buffering time (s) - Average re-buffering time (s)

		- Total re-buffering time (s) - Re-buffering Index Re-buffering duration is calculated on by capturing the Jasvascript events `onPlayerStateChange == PlayerState.BUFFERING` and `onPlayerStateChange == PlayerState.PLAYING.` The Re-buffering Index is computed from the number of re-buferings, the re-bufferings duration and the video's length. Possible values: 0 (good) to infinite (bad).
Video Quality Distribution	-	The playback quality of the video. The following KPIs are derived from a video playback: - First / Last Video Quality - % of Time in each Quality (histogram) - Most Used Video Quality - Average Video Quality[3]
Playtime	s	Over time representation of the video playback. X-Axis: Actual Time Y-Axis: Playback time

5.2.6 Spotify™

The AT4-Agent (section 5.2) forces the mobile device to reproduce an audio track from Spotify™ and measures the quality of experience of the audio streaming.

1. AT4-Agent uses Spotify API to start a music track
2. AT4-Agent sets the playback rate with this API function:

   `public voidsetPlaybackBitrate`

   Spotify API: "*Set the bitrate of the player to specified value. This will take effect for the next chunk of audio that is streamed from the backend. The format or sample rate of the audio data does not change*"
3. Whenever the player state changes, the AT4-Agent captures the event calls: PLAY, TRACK END, etc.
4. Then, while the music track is playing on the device, the AT4-Agent captures the following event calls to measure the KPIs:

   `int onAudioDataDelivered`

   Spotify API: *"Called whenever the player receives audio data. The method is synchronous and therefore blocking*"
5. Based on the API event calls presented above, the AT4-Agent calculates the following KPIs:
   a. Initial Buffering (s): First `onAudioDataDelivered` event - PLAY

[3] Average Video Quality = (% time in 120p * 120 + % time in 240p ... + % time in 4k * 4k) /100.

i. If either of the events is not captured, the KPI is reported as null (absence of measurement).

b. Number of Re-bufferings:

   i. Let T be the time between consecutive `onAudioDataDelivered` events. AT4-Agent counts a re-buffering observation whenever T is longer than 200 ms. Thus, Re-buffering Time is given by T – 40 ms.

c. Total Re-buffering Time (s): Sum of (Re-buffering Time observations)

d. Playback Duration (s): TRACK END – PLAY.

e. Re-buffering Index: Total Rebuffering Time / (TRACK END - First `onAudioDataDelivered` event).

Table 11 summarizes the KPIs reported by the Triangle Testing Framework release 1 for the reference App Spotify^{TM}.

Table 11 Spotify^{TM} KPIs

KPI	**Units**	**Description**
Initial buffering	s	The period between the starting time of loading the audio track and the starting time of playing it.
Re-bufferings	-	When the buffered audio data decreases to a low value, the playback will pause, and the player will enter into a re-buffering state. This KPI shows how many times this event happens. The following KPIs are derived from all the re-bufferings in a audio playback: Maximum re-buffering time (s) Average re-buffering time (s) Total re-buffering time (s)
Track Size	MB	The amount of data downloaded by the device to play the audio regardless of the bit rate used.
Re-buffering index	-	This KPI is computed from the number of re-buferings, the re-bufferings duration and the audio's length. Possible values: 0 (good) to infinite (bad).
MOS	-	Estimation of the audio quality as perceived by a user. Possible values: 1 (bad) to 5 (good). This estimation is based on the initial buffering, and the re-buffering index.
Playback Duration	s	This is the actual duration of the playback from loading the player until the audio track ends.

5.2.7 Facebook^{TM}

The AT4-Agent (section 5.2) forces the mobile device to perform Facebook operations and measures the quality of experience.

This measurement uses the Android Facebook API and uses Facebook test accounts which must be created by the user only once before running the first test on that device.

Table 12 summarizes the KPIs reported by the Triangle Testing Framework release 1 for the reference App Facebook^{TM}.

Table 12 Facebook^{TM} KPIs

KPI	Units	Description
Time to post a comment	s	Elapsed time since users click "post a comment" and the comment is posted in their Facebook account.
Time to post a image	s	Elapsed time since users click "post an image" and the comment is posted in their Facebook account.
Time to post a video	s	Elapsed time since users click "post a video" and the comment is posted in their Facebook account.
Operations successful rate	%	Percentage of operations successfully completed.

5.2.8 Web Browsing

The AT4-Agent (section 5.2) forces the mobile device to download a web page from a destination web server (target IP server) for each test measurement.

A Web browsing session consists of a number of individual web page downloads. The following figure shows a Web Browsing iteration identifying the Setup and Session Time.

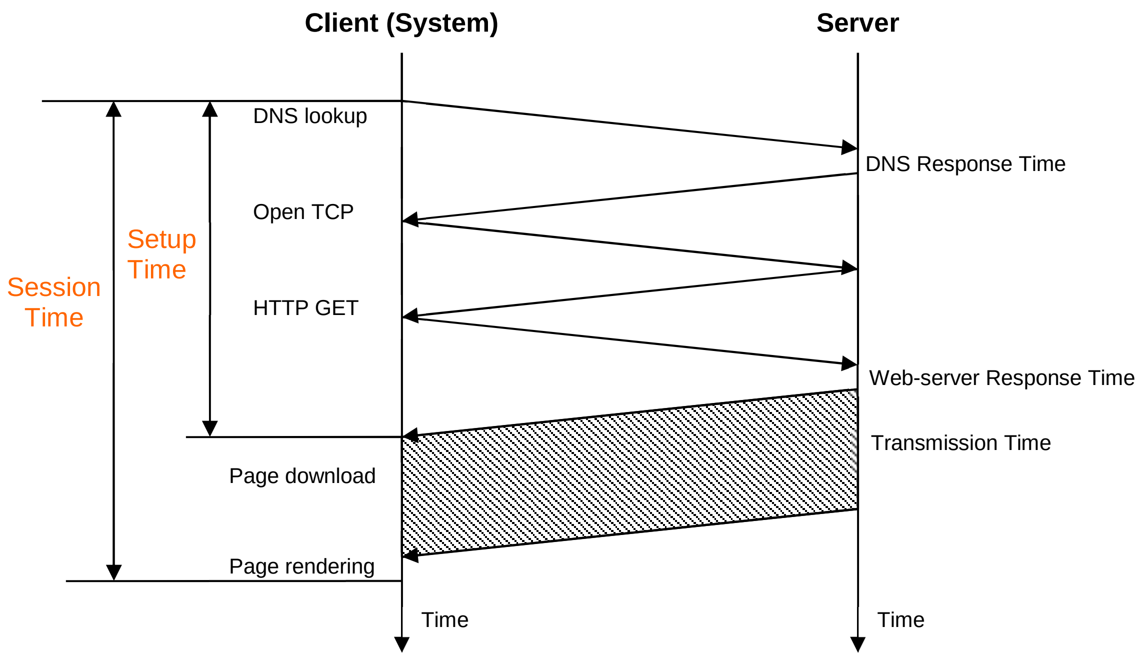


Figure 18 Web Browsing measurements

Table 13 summarizes the KPIs reported by the Triangle Testing Framework release 1 for the reference App Web Browser.

Table 13 Web Browsing KPIs

KPI	**Units**	**Description**
Setup time	s	Time period needed to access the web page, from user entering the URL and hitting “Return”, to the point of time to receive the first byte of the web page
Session time	s	Time period needed to successfully complete the data transfer, from user entering the URL and hitting “Return”, to the point of time to receive the last byte of the web page
Mean data rate	Mbit/s	The average data transfer rate measured throughout the entire connect time to the service. The data transfer shall be successfully terminated

5.2.9 File Transfer

The AT4-Agent (section 5.2) forces the mobile device to transfer (download or upload) a data file from a destination file storage server (target IP server) for each test measurement.

In both the download and upload modes the test uses a single TCP connection to perform the transfer.

Table 14 summarizes the KPIs reported by the Triangle Testing Framework release 1 for the reference App File Transfer.

Table 14 File Transfer KPIs

KPI	**Units**	**Description**
Setup time	s	Time period needed to access the data service, from user entering the URL and hitting “Return”, to the point of time to receive the first byte of the file
Session time	s	Time period needed to successfully complete the data transfer, from the point of time to receive the first byte of the data file, to the point of time to receive the last byte
Mean data rate	Mbit/s	The average data transfer rate measured throughout the entire data file transfer. The data transfer shall be successfully terminated

5.3 Power analyzer

The Keysight N6705B DC power analyzer 4-slot mainframe holds up to 600 W of total power. The N6705B is a highly integrated instrument that combines up to four advanced DC power supplies, digital multi-meter (DMM), oscilloscope, arbitrary waveform generator and data logger. It provides an interface, with all sourcing and measuring functions available from the front panel. In addition, the instrument can be remotely controlled. The aforementioned power analyzer is able to measure the current going into the the UE. The device will support IEEE 1588 for time synchronization between the eNodeB emulator and the power analyzer. This section provides a brief overview of the capabilities of the power analyzer. For a more exhaustive view of the commands the reader is referred to the user manual [3].

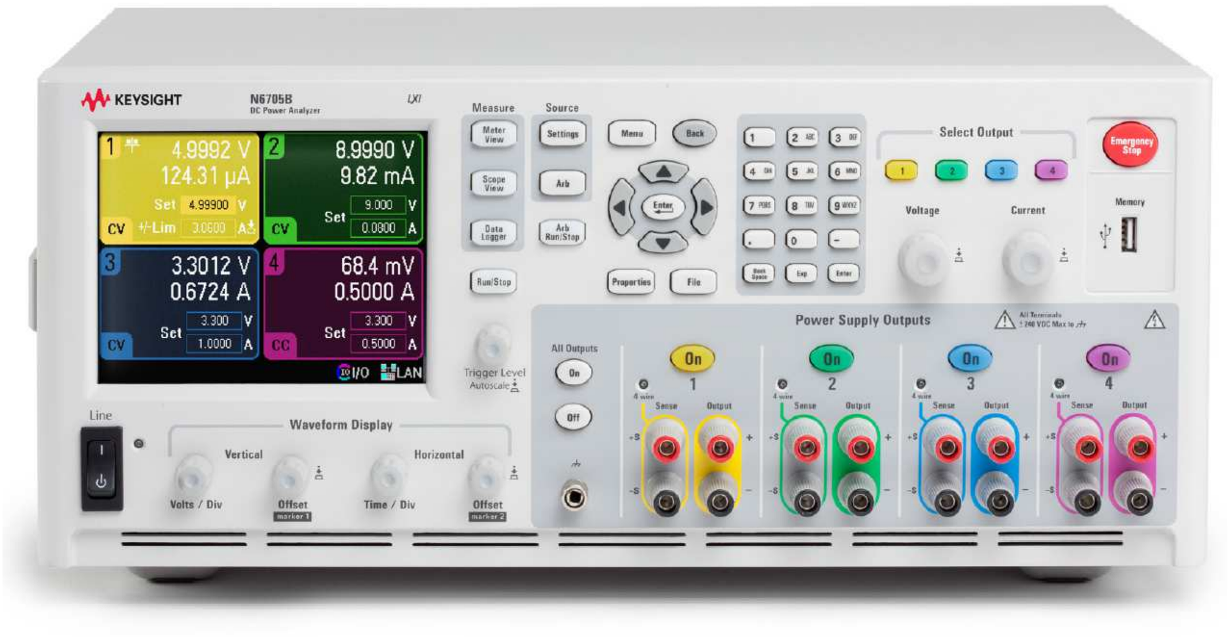


Figure 19 Keysight N6705B DC Power Analyzer

5.3.1 Voltmeter/Ammeter: Meter View

Each DC power module in the Keysight N6705 DC power analyzer has a fully integrated voltmeter and amperimeter to measure the actual voltage and current being sourced out of the DC output into the UE.

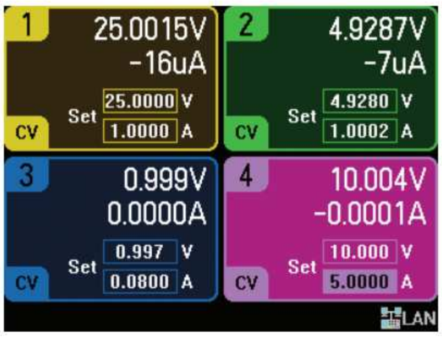


Figure 20 Meter View; all 4 outputs can be viewed simultaneously

5.3.2 Oscilloscope: Scope View

Each DC power module in the Keysight N6705 DC power analyzer has a fully integrated digitizer to capture the actual voltage-versus-time and current-versus-time being sourced from the DC output into the UEDUT. The digitized data appears on the large colour display just like an Oscilloscope.

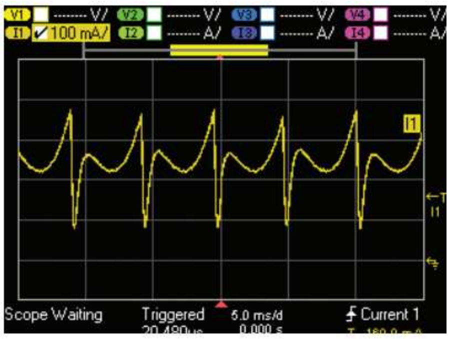


Figure 21 Scope View; voltage and current traces are displayed

5.3.3 Data Logger View

The Keysight N6705 DC power analyzer can also function as a data logger. Using the measurement capability built into each DC power module, the N6705 can continuously log data to the large colour display and to a file. Data can be simultaneously logged on all four DC outputs.

The following table summarizes the data logging specifications:

Table 15 N6705 DC power analyzer data logging specifications

	Standard data logging	Continuous data logging
Sample interval range	75 milliseconds to 60 seconds	20* µs to 60 s *Add 20 µs for each additional parameter (Voltage, Current, Min, or Max)
Sample rate	50 kHz	50 kHz

5.3.4 Control and Analysis Software

The software for the DC power analyzer complements the front panel of the N6705B mainframe, offering advanced functionality and PC control as shown in Figure 22. One showcase application is battery drain analysis for IoT devices.

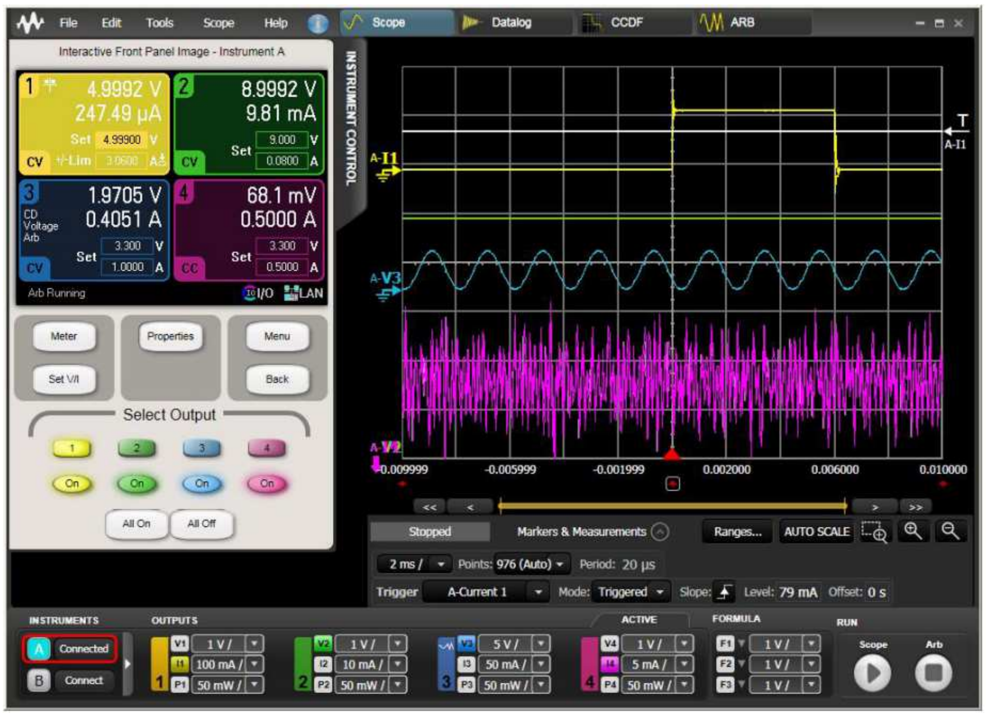


Figure 22 Control and Analysis Software for N6705B Screenshot

5.4 Apps instrumentation

The Triangle testbed will provide several means for extracting information needed to calculate KPIs in apps. Some KPIs can be calculated by measuring the time between two UI events or actions, e.g. the presence of a specific element in the screen, or when a certain user action is performed in the context of an app user flow.

However, to compute KPIs associated to internals actions in the apps, the apps will have to be "instrumented" to provide the required information. This app instrumentation will have to be done by the App developer before submitting the app to the Triangle testbed for testing. This app instrumentation must be both easy to use and lightweight enough, to avoid interfering with the measurements. For the first release of the testbed, the focus has been on providing instrumentation support for Android apps.

5.4.1 Instrumentation library for Android

For Android apps, measurements will be written to the system-wide log, called logcat [10]. All apps running in an Android device can write to this log, and the messages will be timestamped on the device. The contents of the logcat can be retrieved online, e.g. while the Android device is connected to a computer through USB, or offline. Messages can be filtered by their tag (set by the app that sends them) and priority (e.g. verbose, info, warning, error).

To univocally identify the data that is being written as part of the data required for a certain KPI, the app will write to the log with a very specific tag and message format. This enables filtering and parsing the required data from the testbed.

Triangle will provide a library that app developers can use to indicate the measurement points inside de application. The measurements collected at these points will be used to calculate the KPIs. This library will take care of giving the proper tag and format to the log messages, with a developer friendly interface. This library will be optimized for speed and memory, e.g. object allocation will be minimized by using static methods and objects whenever possible.

This library is still under development for the first release of the testbed, and is expected to be ready for the first Open Call experimenters.

The library will provide a set of abstract classes in the Triangle portal (eu.triangle_project.instrumentation.kpis) package, that will enable app developers to provide measurements. To make their intended usage clearer for app developers, the measurements will be grouped into classes, and organized into sub-packages, according to the KPIs which require them. The measurements required by a KPI will be represented by a single abstract class. A KPI will typically require more than one measurement, which we call KPI measurement points in Triangle. Each class provides static methods that must be called by the developer to provide data for a specific measurement point. These methods can take zero or more parameters, depending on what data must be provided for each measurement point. A method with no parameters can also output useful information, such as the presence of an event and its timestamp.

For instance, a KPI that involves measuring the time to download content from the app servers would be supported by a DownloadContentTime abstract class. This KPI may have three measurement points: the content size, and time when the download started and finished. Therefore, this class will provide three methods to provide this information, that the developer will have to call from the app code when the appropriate condition is met. In addition, there are overloaded versions of the methods that indicate the start or end of the download, with an additional timestamp parameter. If no timestamp is given, it will be calculated when the method is called. The explicit version is useful if the developer must obtain the required timestamp from a particular source.

Figure 23 shows how the KPI classes will be organized for the user experience KPIs that were identified in deliverable D2.1, section 5.1.4. The KPIs for each of the app categories are contained in separate packages, with a “shared” package containing KPI classes that can be applied to more than one category.

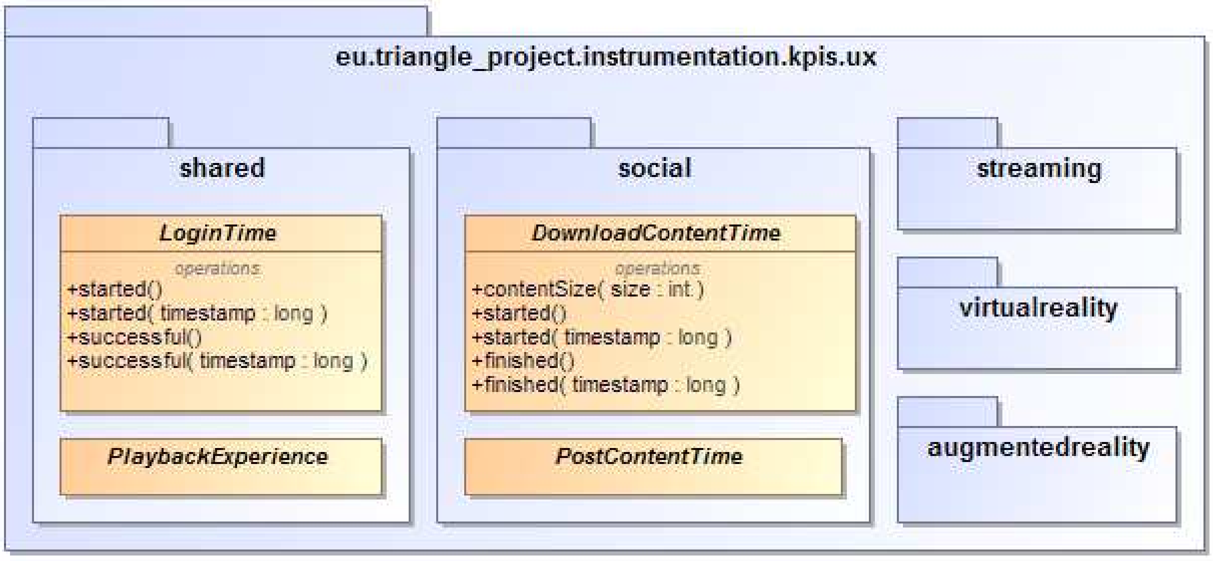

eu.triangle_project.instrumentation.kpis.ux
shared
LoginTime
operations
+started()
+started(timestamp : long)
+successful()
+successful(timestamp : long)
PlaybackExperience
social
DownloadContentTime
operations
+contentSize(size : int)
+started()
+started(timestamp : long)
+finished()
+finished(timestamp : long)
PostContentTime
streaming
virtualreality
augmentedreality

Figure 23 Overview of organization of instrumentation classes for a subset of KPIs

5.4.2 TAP support

A TAP plugin has been designed to extract measurement data written by the instrumentation library, and make it available as a result. Together with the OML plugin for TAP, this will enable cross referencing KPI data from the instrumentation with data from other tools, in the same OML database.

Table 16 shows the instruments that will be provided by the first release of the TAP plugin. A generic IDutLogParserInstrument interface will define common methods to extract measurement data from DUT logs. This will be implemented for each supported OS, e.g. LogcatParserInstrument for Android

Table 16 Measurement parsing TAP plugin instruments

Instrument	**Setting**	**Description**
(IDutLogParser-Instrument)		Interface that must be implemented by instruments that support parsing measurement data from the log of a DUT.
Logcat Parser *(LogcarParser-Instrument)*		Instrument that implements IDutLogParser for parsing measurements from Android's logcat.
	ADB / ADB path	Path to the adb executable in the local file system, that will be used to interact with the logcat.

Table 17 shows the main test steps that will be provided by the TAP plugin. The two main steps are setting up which measurements will be parsed, and performing the actual parsing. By default, all measurements will be monitored, but the subset that is required for a particular KPI can be selected. The ParseDutLogStep performs the actual log parsing for the previously configured measurements.

Table 17 Measurement parsing TAP plugin test steps

Test step	**Setting**	**Description**
Setup measurement parsing *(MeasurementPasrsing-SetupStep)*		Select which measurements will be parsed from the log of a DUT, through an IDutLogParserInstrument.
	DUT / DUT log parser	Select the IDutLogParserInstrument that will be used to parse the DUT log.
	DUT / Device ID	ID of the device whose log will be parsed.
	Measurements / Parse all	If enabled, the test step will try to detect and parse all possible measurements that can be generated by the instrumentation library. If disabled, select a KPI whose measurement points will be parsed in the next setting.
	Measurements / KPI	Select a KPI whose measurement points will be parsed.
Parse DUT log *(ParseDutLogStep)*		Perform the actual parsing of the log of a DUT, to extract the previously configured measurements.
	DUT / DUT log parser	Select the IDutLogParserInstrument that will be used to parse the DUT log.
	Action / Action	Possible values: Activate, Deactivate, Single Shot. If Single Shot is selected, the current contents of the log will be examined to parse the measurements. The other two values can be used to start log parsing in the background, so that other test steps

		can be executed in between.
Setup Logcat regex parsing *(LogcatRegexParsing-SetupStep)*		Configures a custom measurement parsing rule that can be used to extract data from logcat with a custom format not part of the standard measurement points.
	DUT / Logcat parser	LogcatParserInstrument that will be used to parse the custom measurements.
	DUT / Device ID	ID of the device whose log will be parsed
	Result / Name	Name of the result that will be published when a match is found.
	Logcat filter / Tag	Filter the contents of the logcat so that only messages with the given tag are considered.
	Logcat filter / Priority	Filter the contents of the logcat so that only messages with the given priority (e.g. verbose, info, error) are considered.
	Regex / Regex	The regular expression (regex) that will try to be matched in the filtered logcat messages. This regex may have a capturing groups. When a message matches the regex, the values of these capturing groups will be published as results, with the names configured below.
	Regex / Result names	Comma separated list of result names to be given to the value from each capture group in a regex match.

Specific steps to perform ad-hoc parsing will be considered. LogcatRegexParsingSetupStep allows the user to parse messages that match a given regular expression. The capture groups of the match will be published as results of the test step, with the names given in the step settings.

5.5 OML

All the control and measurement tools used during a test may generate significant amounts of data and results. This data is useful for aggregating different measurements and events, and measuring KPIs that span across tools.

5.5.1 OML architecture

The Triangle testbed uses the OML measurement framework [3] to centralize the collection of measurements and other data generated during the test. OML follows a client-server architecture, where several OML clients collect measurements and send them to one or more OML servers using a custom OML protocol. OML clients can use one of the existing client libraries for several programming languages, instead of doing their own protocol implementation.

The measurements from an OML client are grouped into measurements points. Data from each measurement point can be filtered at the client, to perform some processing before forwarding the measurements.

In the Triangle testbed, all measurements are sent to a central OML server, which collects and stores them. This OML server uses a PostgreSQL database server as a backend to store the measurements, not related to the database managed by TAP. The OML server stores the data from each test in a separate database, in the same PostgreSQL database server. Before starting an test, OML clients must be configured with the same test ID, so that the OML server stores all their measurements in the same database.

The data from each measurement point is stored in a separate table in the database. Measurement points produce data as key-value pairs. The measurement point table will contain one column per each key used in the measurements, plus additional metadata, such as the ID of the client, and timestamps at both the client and the server. Each measurement sent by the client will be stored in a row on the corresponding table. The same measurement point ID can be used by more than one OML client. The measurements sent from the two clients will be stored on the same table, but they can be distinguished by the client ID stored on each row. Two other tables will be created with additional metadata per test.

Figure 24 shows the high-level architecture of the OML framework with an example. Two OML clients are sending measurements to a single OML server. The measurements from one client are grouped into two separate measurement points (MP1 and MP2). All the measurements can be filtered before they are actually sent to the OML server. The measurements from each test will be stored in a separate database, in the same PostgreSQL database server.

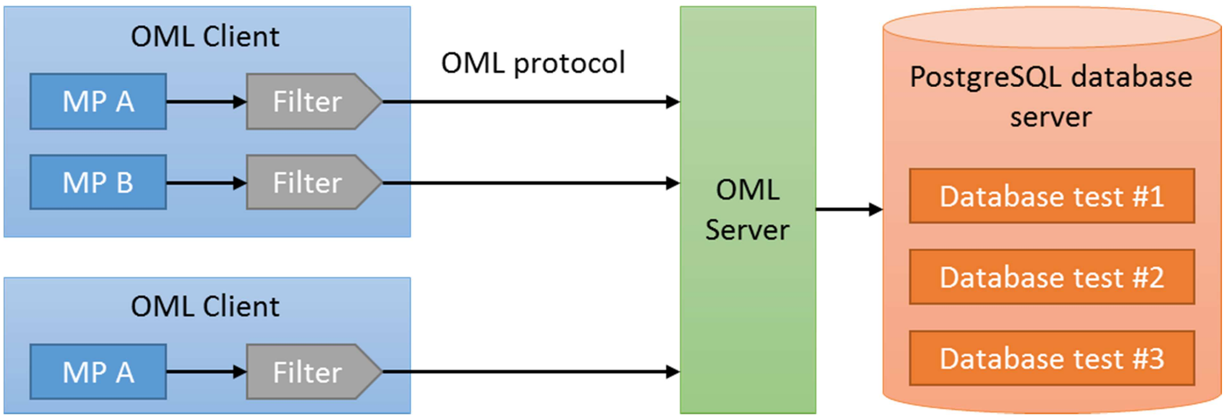
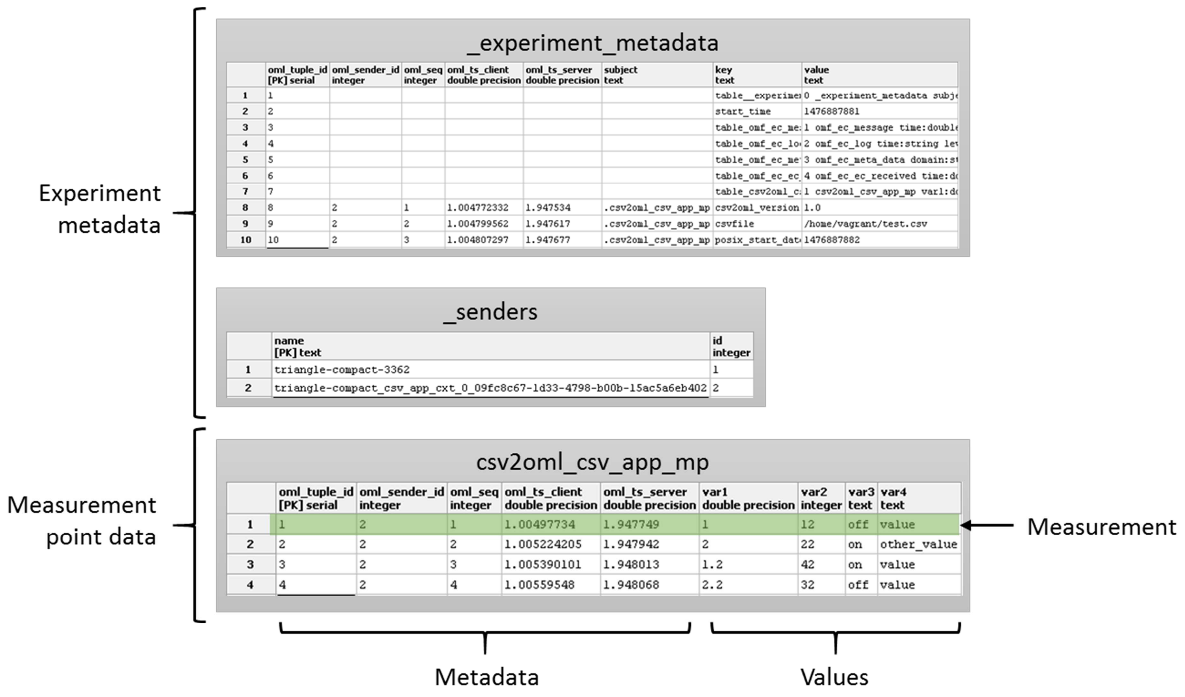


Figure 24 OML architecture

Figure 25 show an example database created by the OML server in the PostgreSQL database server for storing measurements from a test. This test contains a single measurement point, called csv2oml_csv_app_mp, whose measurements are stored in a table of the same name. The first five columns of that table store metadata associated with each measurement, while the other four store the actual values of each measurement. All the measurements have been written by the same client, whose ID is shown in the oml_sender_id column.



_experiment_metadata

	oml_tuple_id [PK] serial	oml_sender_id integer	oml_seq integer	oml_ts_client double precision	oml_ts_server double precision	subject text	key text	value text
1	1						table__experimer	0 _experiment_metadata subje
2	2						start_time	1476887881
3	3						table_omf_ec_me:	1 omf_ec_message time:double
4	4						table_omf_ec_lo	2 omf_ec_log time:string lev
5	5						table_omf_ec_me	3 omf_ec_meta_data domain:st
6	6						table_omf_ec_ec	4 omf_ec_ec_received time:dc
7	7						table_csv2oml_c:	1 csv2oml_csv_app_mp var1:dc
8	8	2	1	1.004772332	1.947534	.csv2oml_csv_app_mp	csv2oml_version	1.0
9	9	2	2	1.004799562	1.947617	.csv2oml_csv_app_mp	csvfile	/home/vagrant/test.csv
10	10	2	3	1.004807297	1.947677	.csv2oml_csv_app_mp	posix_start_dat	1476887882

_senders

	name [PK] text	id integer
1	triangle-compact-3362	1
2	triangle-compact_csv_app_cxt_0_09fc8c67-1d33-4798-b00b-15ac5a6eb402	2

csv2oml_csv_app_mp

	oml_tuple_id [PK] serial	oml_sender_id integer	oml_seq integer	oml_ts_client double precision	oml_ts_server double precision	var1 double precision	var2 integer	var3 text	var4 text
1	1	2	1	1.00497734	1.947749	1	12	off	value
2	2	2	2	1.005224205	1.947942	2	22	on	other_value
3	3	2	3	1.005390101	1.948013	1.2	42	on	value
4	4	2	4	1.00559548	1.948068	2.2	32	off	value

Figure 25 OML database for an experiment with a single measurement point

OML also provides utilities to share data collected by tools that do not implement OML functionality. The csv2oml command line tool sends the contents of a CSV file to an OML server, using the headers of each column as the name of each measurement component.

5.5.2 TAP support

OML measurements have to be configured and performed as part of a TAP test plan that performs a test. This section describes the design for a TAP plugin to support OML measurements in TAP. This plugin includes several test steps and instruments, as usual.

Table 18 shows the instrument that will be provided by the TAP plugin. This instrument will handle the connection to an OML server, and thus the user needs to provide the host and port where that server is running. The server will keep track of the test ID, so that the measurements collected during the execution of a TAP test plan are stored in the same database. It also shows an interface that has been defined for new instruments: ICsvInstrument, Instruments that implement this interface declare that they produce a CSV file as a result of their normal operation. The interface declares only one property, with the path to the CSV file that will be generated.

Table 18 OML TAP plugin instruments

Instrument	Setting	Description
OML Server (OmlServerInstrument)		Handles connection to OML server. Keeps track of the test ID, so that measurements are stored on the same database.
	Server / Host	Host name or IP of OML server.
	Server / Port	Port of OML server.
(ICsvInstrument)		Interface for new TAP instruments that provide CSV files. Instruments that implement it,

Table 19 shows the two test steps that will be provided in the TAP plugin. OmlSetupStep performs basic OML configuration. SendCsvStep sends the rows of a CSV file as individual measurements to the OML server. This file can be one found on the local filesystem, or one provided by an instrument that implements the ICsvInterface. This step will be helpful to add support for tools that generate CSV files as results, without having to implement a full TAP plugin.

Table 19 OML TAP plugin test steps

Test step	**Setting**	**Description**
OML Setup (OmlSetupStep)		Performs initial setup for the OML measurement collection.
	OML / Server	OmlServerInstrument which will be set up.
	Test / Custom name	If enabled, the user can set a custom name for the new test, which will be the name of the database where measurements will be stored. If disabled, a unique test name will be automatically generated.
Send CSV to OML (SendCsvStep)		Sends the contents of a CSV file to an OML server.
	OML / Server	OmlServerInstrument to which measurements will be sent.
	CSV / Source	Selection between: File, Instrument. Select the source of the CSV file: a file in the file system, or a CSV generated from a compatible ICsvInstrument.
	CSV / File path	Path to a CSV file in the file system, whose contents will be sent to the OML server.
	CSV / Instrument	An ICsvInstrument that will provide the CSV file that will be sent to the OML server.
	CSV / Has Headers	If true, the first line of the CSV file will be interpreted as a header. The values of this header will be used as the names for each column. If false, column names will be assigned automatically.
	Measurements / Measurement point	Name of the measurement point defined for this CSV file.

Finally, the plugin will also provide a TAP result listener that sends the results published by TAP test steps to an OML server, shown in Table 20. A result listener is the best option for publishing results from plugins which follow TAP conventions and publish results through the TAP API.

Table 20 OML TAP plugin result listeners

Result listener	**Setting**	**Description**
OML Result Listener		Result listener that sends the results received from

(OmlResultListener)		TAP test steps to an OML server. The name of the ResultTable published by a test step will be used as the measurement point name.
	OML / Server	OmlServerInstrument to which measurements will be sent.

Figure 26 shows the main classes that will be part of the implementation of the TAP plugin for OML. Only one instrument has been defined, OmlServerInstrument, to represent the connection to a particular OML server. This instrument will hold the name of the test, which will be used to identify the database where all measurements of a TAP test plan execution will be sent to.

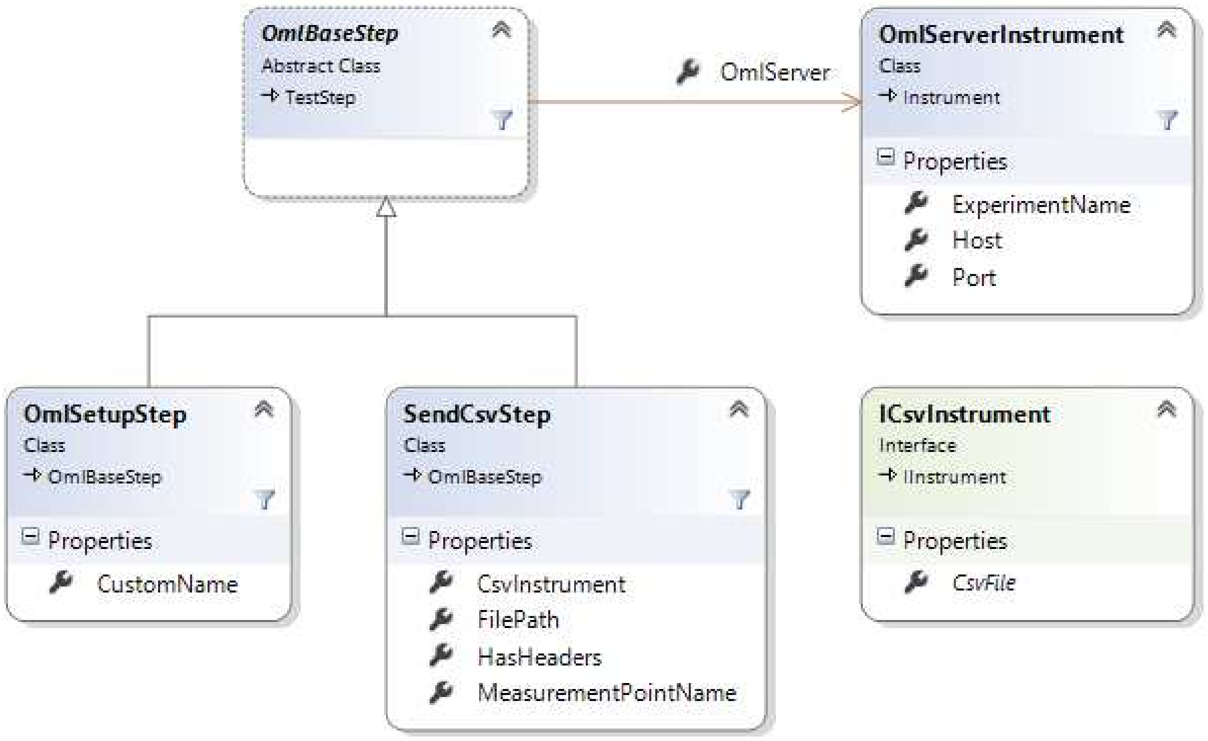
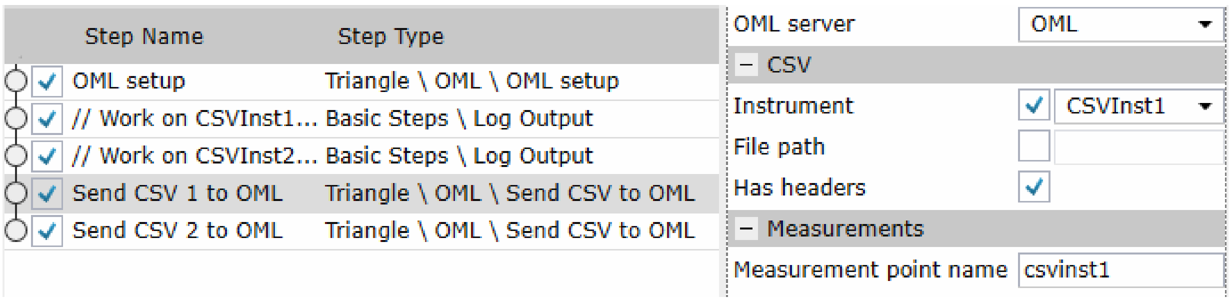


Figure 26 Main classes of TAP plugin for OML

Figure 27 shows an example of a TAP test plan using the test steps and instruments proposed for the OML TAP plugin. The test plan uses an OmlServerInstrument, which will collect measurements, and two instruments that implement the ICsvInstrument interface. The first step is to configure the name of the OML test, which is followed by the steps that make use of both ICsvInstruments. It finalizes with two test steps that send the CSV files generated by both instruments. The Instrument selector allows the user to select only those instruments that implement the required interface.



	Step Name	Step Type
✓	OML setup	Triangle \ OML \ OML setup
✓	// Work on CSVInst1…	Basic Steps \ Log Output
✓	// Work on CSVInst2…	Basic Steps \ Log Output
✓	Send CSV 1 to OML	Triangle \ OML \ Send CSV to OML
✓	Send CSV 2 to OML	Triangle \ OML \ Send CSV to OML

OML server		OML
− CSV		
Instrument	✓	CSVInst1
File path		
Has headers	✓	
− Measurements		
Measurement point name		csvinst1

Figure 27 Example of TAP using OML TAP plugin

6 RAN (Radio Access Network)

This first version of the Triangle testbed has integrated the UXM Wireless Test Set from Keysight as the radio access network (RAN) of the testbed. The UXM is a flagship mobile network emulator that provides state of the art test features.

6.1 eNodeB emulator

The Keysight E7515A UXM Wireless Test set is capable of emulating a 2G, 3G, 4G and NB-IOT base stations. The UXM is a highly-integrated instrument created for functional and RF design validation in 4G and NB-IOT. It provides the integrated capabilities needed to test the newest designs, delivering LTE-Advanced Pro data rates up to 1 Gbps. In addition, the UXM allows functional test by emulating a wide range of complex network operations, such as LTE intra-RAT and LTE inter-RAT mobility, WLAN offload and end to end VoLTE. The UXM also provides protocol messaging.

The number of features and capabilities of the UXM is very extensive. The purpose of this section is to highlight its most relevant features for the testbed. For a complete reference of capabilities and documentation about such capabilities, the reader is referred to [17] [18].

Feature	**Description**
Basic Cell Configuration	The UXM provides an API to control most of the network parameters. Among the basic aspects that can be configured, the following are highlighted: - Duplex mode: FDD or TDD - Downlink and Uplink bandwidth - Most of the FDD and TDD LTE bands are supported - TDD frame configuration
RF Channel Conditions	The UXM is capable of emulating different channel conditions in terms of signal levels, fading profiles, and noise and interference profiles. The following propagation conditions are supported: - Static - Extended Pedestrian A (EPA) - Extended Vehicular A (EVA) - Extended Typical Urban Model
Mobility	The UXM is capable of emulating the following mobility scenarios: - Intra-System Handover - Inter-System Handover - Roaming
Cell Load	The UXM is capable of emulating different cell loads by controlling the number of resources allocated to the user.
Dual Connectivity	Within the scope of the Triangle project the Testing framework shall be able to support dual connectivity between 3GPP radio access nodes and between 3GPP and non-3GPP radio access nodes.
Supported Formats	GSM/UMTS/LTE/LTE-A/NBIOT

6.2 RF switches

Triangle expects to increase the number of reference devices in the testbed. Moreover, Triangle users might select a given device at any moment. Therefore, these devices should be available to the testbed at all times. This poses a considerable challenge as the RAN emulator only has 4 RF connectors. Obviously, wiring all the devices to the RF ports of the RAN emulator could create instabilities, e.g., one rogue device connecting to the network instead of the desired one. To avoid such issues and for the sake of a more stable system Triangle uses an RF switch. This switch is cable of creating a 1 to 1 mapping, meaning that a given time only a device is connected to the RAN emulator. Triangle is using the Keysight L7104A component, which is an electro-mechanical switch providing isolation and 0.03 dB insertion loss repeatability. The RF switch is shown in Figure 28.

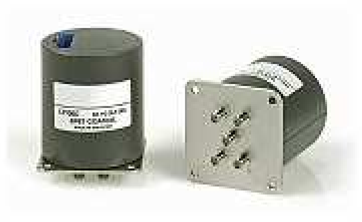


Figure 28 Keysight RF Switch Product Number L7104A

6.2.1 RF Switch Controller

The RF switch is controlled by a LXI-compliant 11713C attenuator/switch driver, which provides remote or front-panel drive control. This controller will be used to configure the RF switch to connect and only connect the desired UE at a given time. The controller is shown in Figure 29.

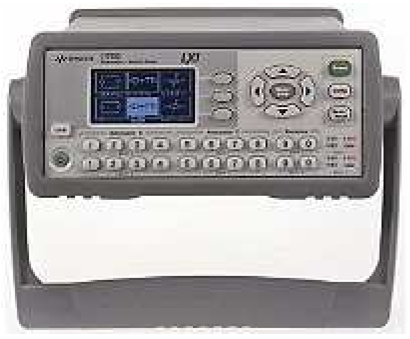


Figure 29 Keysight 11713C LXI-Compliant Attenuator/Switch Driver

7 EPC (Evolved Packet Core)

The core network available in the Triangle testbed is provided by Polaris Networks. It is a core network emulator but it provides carrier grade performance for up to 2000 clients. The platform supports positive and negative behaviours, remote sniffing of all the interfaces of the network elements, introduction of traffic impairments in the transport interfaces and support for multiple instances of the following elements:

- MME
- SGW
- PGW
- PCRF
- HSS
- ePDG
- ANDSF

The configuration and creation of these components can be done using a GUI interface, accessing a programmable API written in TCL, and partially with a C++ API developed in the FLEX project [19]. The list of configuration and measurement parameters are described in the following sections.

The integration of the EPC is still under discussion. The first release of the testbed is going to offer a static scenario, and for the next versions the consortium is discussing what and how to offer to experimenters.

7.1 Configuration parameters

Table 21 MME configuration parameters

MME Features	**Details**
Standard Interfaces	- S1-MME, S11, S10, S6a, S13, S3, SBc, SGs, S102, M3, Sm
Protocols/Interfaces with Configurable Behaviour	- S1 Setup, S1AP, NAS, GTv2, SGsAP, Diameter
Procedures	- Attach, Paging, SMS, Detach, Location Report, Handover, etc.
Capabilities	Partial Path Failure, PGW Restart Notification, Modify Access Bearer Request, Network Triggered Service Restoration, Relay eNB, IMS Voice over PS session, Diameter Proxy Agent Type 2, Extensive Diameter Validation, IPv6/IPv4, Roaming, Configuration of protocol policies.
S1AP Procedures with Configurable Behaviour	Setup Request, Reset, eNB Configuration Update, eNB Configuration Transfer, Initial UE Message, Uplink NAS Transport, UE Context Release Request, Handover Required, Handover Notify, Handover Cancel, Path Switch Request, Initial Context Setup Response, Handover Request ACK, E-RAB Setup Response, E-RAB Modify Response, E-RAB Release Response, UE Context Modification Response
NAS (EMM and ESM) Procedures with	Attach Request/Complete, Detach Request/Accept, TA Update Request/Complete, Security Mode Complete/Reject, Service Request, Authentication Response/Failure, Activate default/dedicated

Configurable Behaviour	EPS bearer accept/reject, Modify EPS bearer context accept/reject, PDN connectivity request
GTPv2c Procedures with Configurable Behaviour	Create/Update/Delete bearer request, identification request/response, context request/response/acknowledge, MBMS session start/update/stop,
SGsAP Procedures with Configurable Behaviour	SGsAP location update ack/reject, SGsAP paging request, SGsAP release request, SGsAP alert request
Diameter Procedures with Configurable Behaviour	Cancel-location-request, insert/delete-subscriber-data-request, reset-request

Table 22 PGW configuration parameter

PGW Features	**Details**
Standard Interfaces	S5-c, S5-u, S8-c, S8-u, SGi, S2a, S6b, Gx
Protocols/Interfaces with Configurable Behaviour	GTPv2-C, PMIPv6, Diameter, GTPv1-U
Functionalities	Traffic generation, traces, apn, protocol configuration
GTPv2C Procedures with Configurable Behaviour	Create Session Request, Delete Session Request, Modify Bearer Request, Modify Bearer Command, Delete Bearer Command, Bearer Resource Command
PMIPv6 Procedures with Configurable Behaviour	Proxy Binding Update, Binding Revocation, Heart Beat
Diameter Procedures with Configurable Behaviour	Re-Auth Request, Session Termination Request
GTPv1U Procedures with Configurable Behaviour	QCI demands.
PGW Features	Details
Standard Interfaces	S5-c, S5-u, S8-c, S8-u, SGi, S2a, S6b, Gx

Table 23 PCRF configuration parameters

PCRF Features	**Details**
Standard Interfaces	Gx, Rx, S9, Gxx
Functionalities	Services Creation/Modification/Deletion, IMS, Roaming
Diameter Procedures with Configurable Behaviour	Authorisation and Authentication Request, Credit Control Request, Session Termination Request

Table 24 SGW configuration parameters

SGW Features	**Details**
Standard Interfaces	S11/S4-c, S1/S4/S12-u, S5/S8-c,S5/S8-u, Gxc
Protocols/Interfaces with Configurable Behaviour	GTPv2-c, PMIPv6, GTPv1-U
Capabilities	Partial Path Failure, PGW Restart Notification, Modify Access Bearer Request, Network Triggered Service Restoration, Diameter Proxy

	Agent Type 2
GTPv2C Procedures with Configurable Behaviour	Create/Delete/Modify/Update Session Request, Release/Modify Access Bearer Request, Create Indirect Data Forwarding Tunnel Request
PMIPv6 Procedures with Configurable Behaviour	Binding Revocation, Heart Beat
GTPv1U Procedures with Configurable Behaviour	QCI demands.

Table 25 HSS configuration parameters

HSS Features	**Details**
Standard Interfaces	S6a, Zh, Cx
AAA interfaces	S6b, STa, SWd, SWm
SPR Functionality	Service definition (Service data flow, piggybacked bearer creation, QCI, Priority Preemption, UL-GBR, UL-MBR, DL-MBR, DL-GBR), AMBR, GBR, Charging, QCI, Priority,
Subscribers	Subscription groups, Subscription Profiles
Diameter Procedures with Configurable Behaviour	User-Authorisation-Request, Server-Assignment-Request, Location-Info-Request, Multimedia-Auth-Request, Authentication-Information-Request, Update-Location-Request, Purge-UE-Request, Notify-Request
HSS Features	Details

Table 26 ePDG configuration parameters

ePDG Features	**Details**
Standard Interfaces	SWu, S2b, SWm
Protocols configuration	IKEv2, PMIP, Diameter
IKEv2 Configurable Behaviour	IKE_SA_INIT, IKE_AUTH, INFORMATIONAL
PMIPv6 Configurable Behaviour	Binding Revocation, Heart Beat
Diameter Configurable Behaviour	Re-Auth Request, Abort Session Request

Table 27 ANDSF configuration parameters

ANDSF Features	**Details**
Standard Interfaces	S14, Zh, Ub
Rules	ISRP Rules (flow based, service based, non-seamless offload rules), ISMP Rules (access networks, time of day conditions, validity areas)
Non 3GPP Access	WLAN, 3GPP2, WiMax, Geo.

ANDSF Features	Details
Standard Interfaces	S14, Zh, Ub

7.2 Measurement and behavior

The EPC deployment can provide the signalling available in between all the components of the network. This feature can be supported by directly sniffing in the interfaces but also with an automatic system provided by the Minimization of Drive Test features (see TS 32.422) that can send the messages in XML format to an external server.

Additionally, the EPC modules themselves can provide statistics regarding the procedures associated with each protocol. All the EPC modules provides information on the received messages as well as failure/success counts for each procedure

Table 28: HSS Statistics

Interface	**Procedures**
S6a/S6d	Update location, canel location, authentication information retrieval, insert subscriber data, delete subscriber data, purge ue, reset, notify and ME identify check.
Cx	User authorization, authentication information retrieval, server assignment, user location information retrieval, registration termination, push profile.
SLh	Routing info.
Sh	User data, profile update, subscribe notifications.
S6b	Authorization, abort session, session termination.
STa and SWm	Authentication Authorization, abort session, session termination.

In the case of the MME the system also provides statistics on the subscribers in the different mobility states and number of connections in the different interfaces. In the case of general procedures there is also some information regarding the minimum, maximum and average time to finish some procedures such as attach, detach, tracking area update, S1-handover, X2-handover, service request, paging, pdn connection, dedicated bearer activation, etc.

Table 29: MME Statistics

Interface	**Procedures**
NAS	Attach, network/UE initiated detach, security, service request, TAU, ESM information, GUTI reallocation, etc.
S1AP	S1 Setup/Reset, eNB configuration update, MME configuration update, initial context setup, UE context modify, UE context release, e-RAB setup, e-RAB modify, e-RAB release, HO preparation, etc.
GTPv2-C	Create session, delete session, modify bearer, release access bearer, delete bearer, create/delete indirect data forwarding tunnel, get identification, create context, forward relocation, forward relocation complete, etc.
Diameter	Update/cancel location, authentication information, insert/delete subscriber data, purge UE, reset, notify, ME identity check, mobile terminated location report.
M3AP	M3 setup, session start/stop.

LCSAP	Location service, reset.
SBcAP	Write replace warning, stop warning.
SGsAP	Paging for non-EPS services, location update for non-EPS services, Non-EPS alter, IMSI Detach from EPS/non-EPS services, VLR/MME/HSS failure, MME information, tunnelling of NAS messages, service request.
S102AP	S102 session establishment/termination, S102 tunnel redirection

The PCRF also provides some timing statistics for procedures such as IP-CAN session establishment/termination, service activation/modification/deletion, S9 sub-session establishment/termination, AF session establishment/termination, gateway control session establishment/termination.

Table 30: PCRF Statistics

Interface	**Procedures**
Gx, Gxx and S9	Credit control, re-authorization
Rx	Authentication authorization, re-authorization, abort session, session termination.

The PGW provide timing stats for PDN connection/disconnection and dedicated bearer activation/modification/deactivation and also statistics regarding the throughput and number of packets in the S5/S8 and S2a/S2b interfaces per user and RAB id.

Table 31: PGW Statistics

Interface	**Procedures**
GTPv2-C	Session creation/deletion, bearer modification, UE requested bearer operation, network initiated dedicated bearer creation/update/deletion.
Gx	Credit control, re-authorization
S6b	Authentication authorization, re-authorization, abort session, session termination.
PMIPv6-C	Create/delete session.

The SGW provides information on the data traffic per user and per bearer in the S1-U/S4-U/S12-U and S5-U/S8-U interfaces. It also provides information on the control packets of the data plane such as echo request/response, error indication, SEH notification or end packet.

Table 32: SGW Statistics

Interface	**Procedures**
GTPv2-C	Create/Delete session, modify bearer, release access bearer, downlink data notification, create/delete indirect data forwarding tunnel, create/update/delete bearer, change notification,

	update/delete PDN connection set, modify access bearer, PGW restart notification.
PMIPv6-C	Create session/delete session.
Diameter	Credit control, re-authorization.

And the testbed can also provide sniffed packets for all the interfaces that interconnect the core network both internally and externally.

8 Transport

8.1 SDN

The transport layer consists of three independent network domains interconnected through a virtualized routing environment and managed by several SDN components. On one hand this approach allows to quickly test new configurations without the time consuming task of routing new physical lines between the networks and, on the other hand, it does not limit the future expansion of the test facilities as the virtualized network equipment can connect to their physical counterpart, and thus any component can be moved to another location and only those routers and switches on the edge have to be made aware of the new addresses. Figure 30 shows the three domains considered owned by two different actors, the mobile operator and the backhaul operator:

- RAN or Access network of the mobile operator, which includes the base stations and the network equipment.
- The EPC of the mobile network.
- The distribution network between the two above.

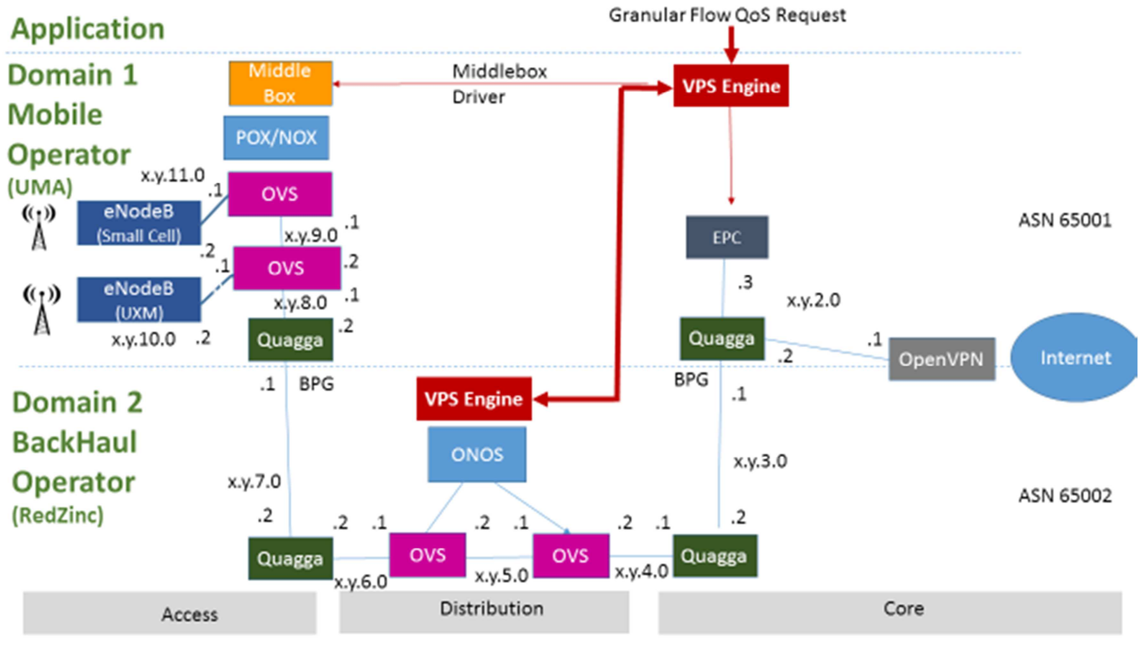


Figure 30 Software defined network deployment at Triangle testbed

The devices inside each domain are isolated from each other, effectively belonging to different networks. The interconnection is provided by the following three components:

- OpenvSwitch, a fully compliant OpenFlow implementation for virtual switches, designed to be run in a virtualized environment but that can also be used with real network equipment.
- Quagga, a software router implementing several routing and discovery algorithms as well as advanced network functionality like multipath Border Gateway Protocol (BGP) or Intermediate-System to Intermediate-System (IS-IS) routing.
- ONOS, a network orchestrator or SDN controller which provides a transparent interface to monitor and manage a distributed deployment in different subnetworks. Different

instances running in different networks can be interconnected to present a unified interface to control and visualize all the resources and traffic flows from a central location. It also provides a high level API to inject new rules for specific links based on the source or target address, or the type of traffic.

The initial deployment consists on Virtual Machines hosted on a Linux KVM hypervisor environment, with isolated virtual networks configured in the host machine. The different subnets are not linked with each other and the host's only role is to provide a gateway to the public internet for the VMs, but it cannot be used to break that isolation. The way to interconnect the networks is the equivalent of a real environment: each router or switch needs an additional port (i.e. a virtualized Ethernet device) for every network it belongs to.

The low level configuration of each VM, that is, the IPs and the number of networks it connects to, is expected to remain stable for the duration of the project so it has been made by hand. The configuration of the transport network is one of the cornerstones of the setup so it has to allow automatization.

Once deployed, the Quagga routers should be able to work without direct intervention from the operator. As it implements discovery algorithms like BGP and OSPF it will detect other routers in the networks it is connected to, and will establish and maintain the routes between the hosts in each network.

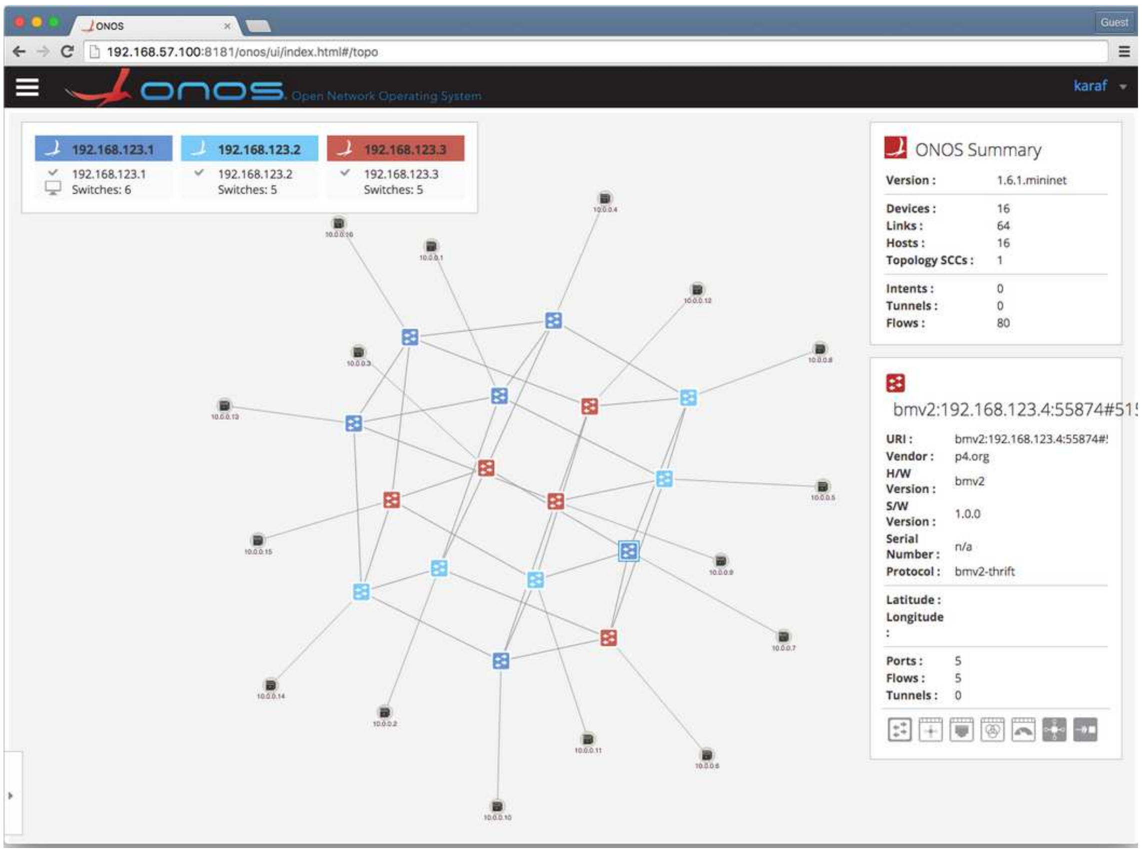
OpenvSwitch works out-of-the-box as a learning switch which makes a discovery search using the ARP protocol when a new host sends packets through it, but it also has advanced features that can be configured using the OpenFlow protocol. Some of the switching capabilities expected to be used in the project are:

- Creation of data plane and control plane as differentiated flows in each switch. It will allow to process both types of traffic with different priorities or even through different routes.
- User traffic identification and matching against rules to provide different QoS to the data coming from the UE.
- Transparent mirroring of certain traffic flows to allow advanced interconnection schemas between the EPC and groups of users.

OpenvSwitch can be configured by command line (or by means of a script) specifying the flow rules to be installed or modified, but this approach can be cumbersome in a dynamic environment as it would be necessary to create the commands on the fly.

Another way to configure the switch is to set up a *controller node* in each OvS instance. This way, when a packet without a matching rule arrives to the switch, it will ask that controller what actions should it perform for that packets and whatever follows in the same flow.

The ONOS orchestrator will take the role of the switch controller, as it provides a high level Java API to inject OpenFlow rules as a result of a petition from any switch or router associated with it. It also provides a web interface to monitor the network and to install and remove rules for different instances of the switches and routers, as can be seen in Figure 31.



ONOS Summary

Version : 1.6.1.mininet

Devices : 16
Links : 64
Hosts : 16
Topology SCCs : 1

Intents : 0
Tunnels : 0
Flows : 80

bmv2:192.168.123.4:55874#51

URI : bmv2:192.168.123.4:55874#
Vendor : p4.org
H/W Version : bmv2
S/W Version : 1.0.0
Serial Number : n/a
Protocol : bmv2-thrift

Latitude :
Longitude :

Ports : 5
Flows : 5
Tunnels : 0

Figure 31 ONOS Interface

8.2 Emulated impairments

The testbed will offer the possibility of integrating artificial impairments in the interfaces of the core network and the application servers. To do so, two different alternatives are being explored: impairments introduced by the EPC emulator and impairments introduced by external applications.

The impairments introduced by the EPC emulator can be set in any of the interfaces of each of the components of the EPC and it enables the definition of the following impairments:

- Limit of packets, which is the number of packets to which the impairment will be applied. If the limit is set to 0, it will be applied to all the packets.
- Delay Parameters, which includes, delay, jitter, delay correlation, delay statistical distribution, percentage or reordered packets, percentage of reorder correlation and gap.
- Loss Parameters, distribution type, percentage of lost packets, percentage of correlation.
- Corruption and Duplication, percentage and correlation for both.

To access this functionality EPC emulator offers a graphical interface and a TCL script.

The other possibility to integrate emulated impairments in interfaces not related to the EPC is the use of external applications. The main explorations by the TRIANGLE project have been the use of mininet and dummynet [1] (the userspace version is able to process 6 million packets per second with simple filtering). The main issue with these types of approaches is the performance that can be offered in real environments, in the preliminary test carried out by the UMA team mininet can offer up to 100 Mbps when connected to equipment outside the emulation environment, and the integration can be done using command line parameters. There are still ongoing tests efforts with dummynet, and the integration with the testbed will be done based on command line parameters.

8.3 Virtual Path Slice Engine

RedZinc's VPS Engine, VELOX, provides an API for 3rd party application developers so that services between two endpoints with a specified bandwidth reservation can be requested without the specific knowledge of how the network itself is deployed or how many Autonomous Systems need to be involved in order to establish the service. The API also allow the listing of running services and services available for request. Unique API keys are generated on demand and bound to developers so that all services can be correctly charged, adding the capability to simulate a financial layer in the testbed.

8.3.1 VPS engine usage

In order to use the VELOX API an application must:

- Create a TCP connection to known IP address/port (provided by local operator)
- Write Request (as a single text line, new line ends a request)
- Read Response (sent as a single line)
- Connections are terminated on the VELOX side after sending the response

All Requests must use the API key generated by the local operator VELOX.

8.3.2 Usage considerations

API Key

The API Key provided will always be a standard UUID in human readable format without dashes.

Example: ECE335024E3E466CA98BF5014D5C7D86

IPv4 vs IPv6

VELOX Supports both IPv4 and IPv6 services, but does not allow IPv4 mixed with IPv6, in requests that have both source and destination addresses, both must be of the same IP version. All versions of IPv6 abbreviation are supported.

Security

Currently the system considers a safe connection already exists between the Operator and the 3rd Party.

8.3.3 Usage example

In this scenario the client application is considered to be any application that uses the VELOX API to access VELOX services.

Modify and Run requests are considered optional since not all circumstances will require their use.

While the List request is the first to be executed it is not mandatory before every Trigger request, the use of the List request should be done as deemed necessary in order to check for any service library changes.

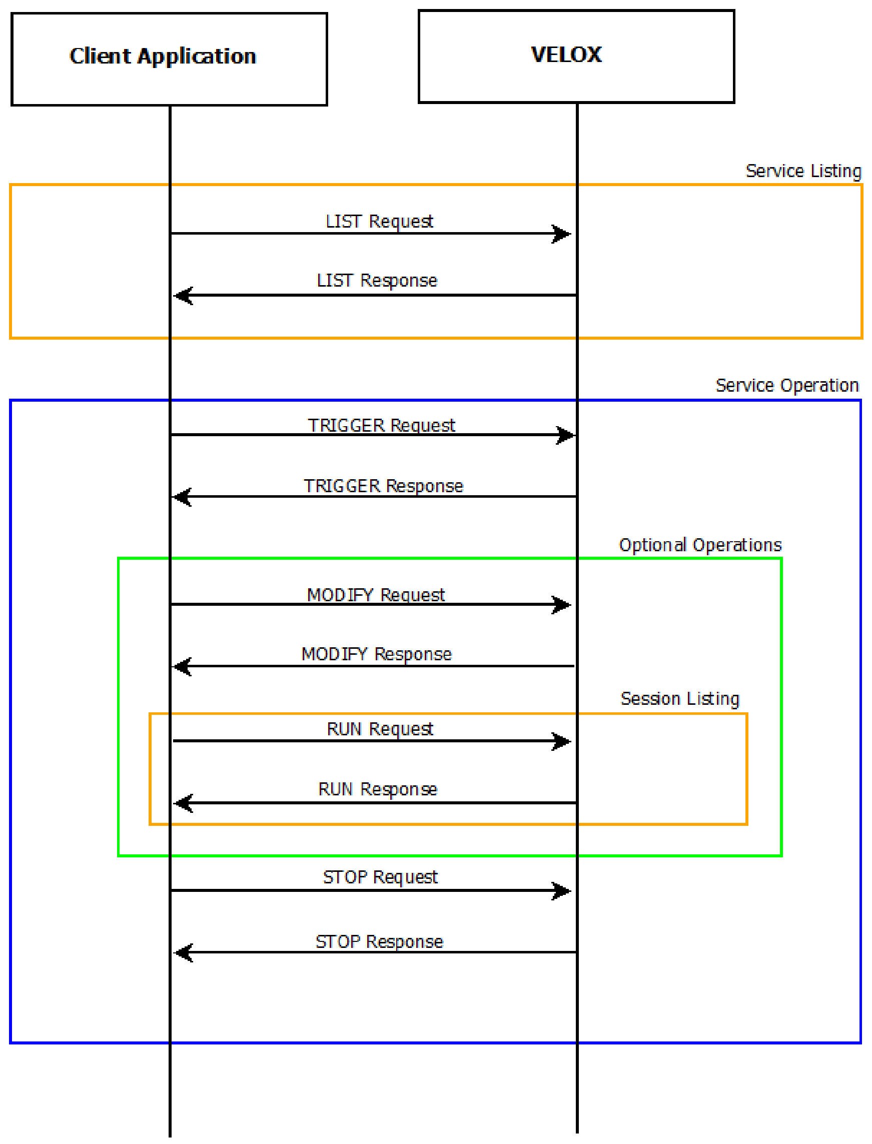


Figure 32 VPS engine usage example

A detailed description of VPS engine API is provided in Appendix 3.

9 UE (User Equipment and accessories)

9.1 Supported UEs

The Triangle testbed provides a set of mobile phones ready to be used. These devices have been connectorized as shown in this section. Devices sent for testing in the Triangle test bed must be provided in pairs, one connectorized. The devices available in the first release of the Triangle testbed for testing Apps are listed in Table 33.

Table 33 List of supported UEs in the Triangle testbed

UE Id or #	**Description**	**LTE Bands**	**UE Category**
DEV #1	Samsung Galaxy S4 (connectorized)	1, 3, 5, 7, 8, 20	3
DEV #2	Samsung Galaxy S5 Neo (connectorized)	1, 3, 5, 7, 8, 20	6
DEV #3	Samsung Galaxy S6 (connectorized)	1, 2, 3, 4, 5, 7, 8, 12, 17, 18, 19, 26, 28	6
DEV#4	Samsung Galaxy S7 (Exynos) (connectorized, only main antenna)	1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 39, 40, 41	9

9.2 RF connection

The UXM Wireless Test Set integrates channel emulation and digital generation of impairments such as AWGN, which is a critical feature to achieve high accuracy when setting SNR conditions. In particular, standard multipath fading profiles defined by 3GPP are supported to emulate reference propagation conditions. In order to preserve the radio conditions configured at the UXM Wireless Test Set the radio connection is conducted through cables and the devices is enclosed in a shielding box as shown in Figure 33.

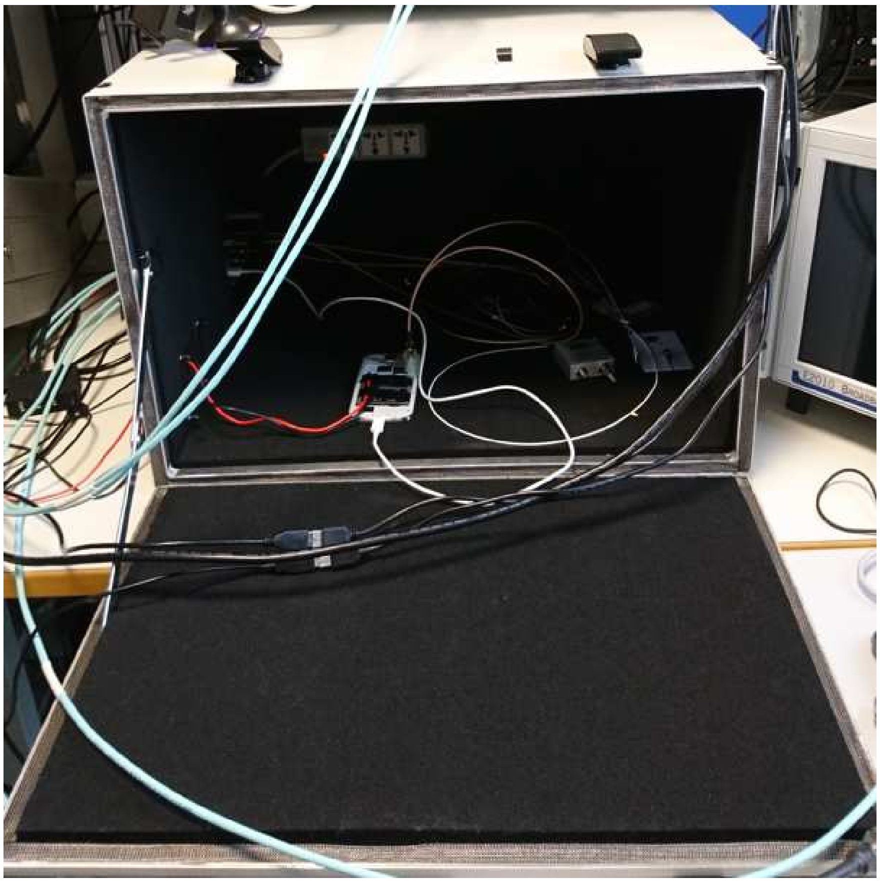


Figure 33 Shielding Box

The following figures show how the antennas on supported UEs have been connectorized with RF clabes, so that they can be connected to the UXM.

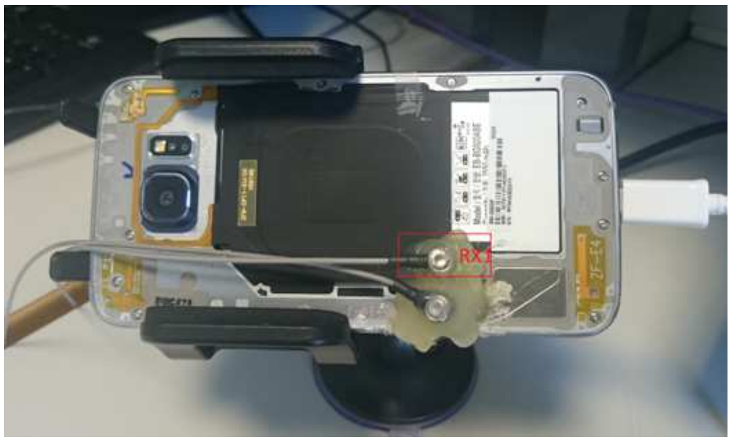


Figure 34 Samsung Galaxy S6 main and diversity internal antennas

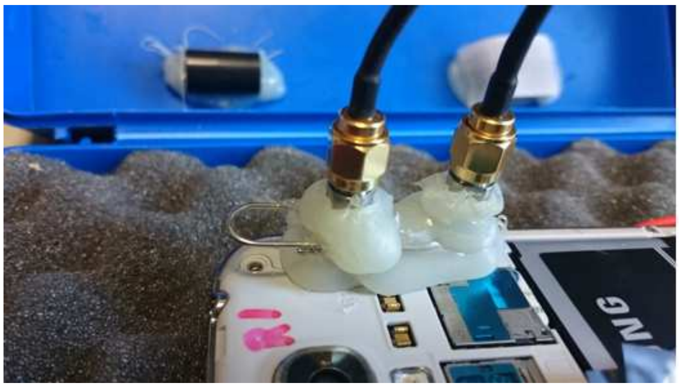


Figure 35 Samsung Galaxy S4 main and diversity internal antennas

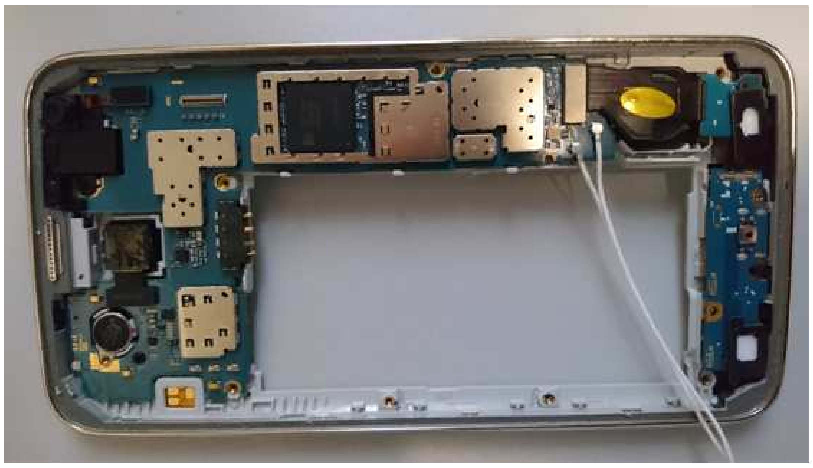


Figure 36 Samsung Galaxy S5 main and diversity internal antennas

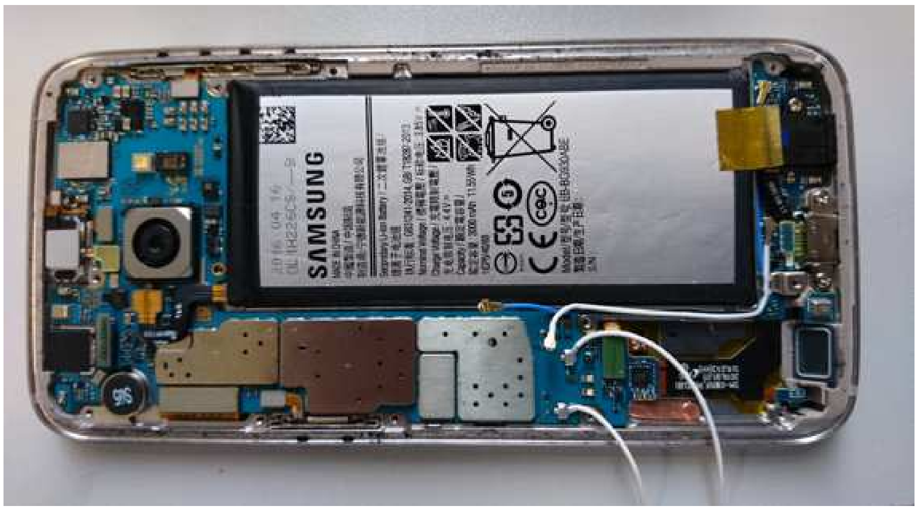


Figure 37 Samsung Galaxy S7 main internal antennas

9.3 DUT HUB

Similarly, there is also a need to connect the UEs via USB ports. This connection is used to send command to the UEs and to operate the applications and services. To avoid the challenges of connecting all the UEs directly, we use a Keysight DUT HUB. This is basically a USB hub which can be controlled via SPCI commands. This allows us to only connect the desired UE to the computer controlling the UE. The concept is very similar to the RF switch explained early.

The DUT features 4 high power USB ports. It is capable of delivering 1.5 A per port (Max. 5 A in total). It is able to power on or power off each USB port separately. The list of commands is provided in the table below:

Table 34 DUT HUB SCPI command reference

Command	**Description**
*ON <port>	Turns on power to a given port identified by <port>. E.g. “*ON 1” to turn on power for port 1.
*OFF<port>	Turns off power to a given port identified by <port>. E.g. “*OFF 1” to turn off power for port 1.
RATE<Hz>	Set the sampling rate to between 20 and 1000 Hz.
READ?	Returns the last acquired samples as a IEEE-488 block, and clears the sample buffer. Samples are encoded as signed 16-bit numbers in bundles of 4 – one value for each port. The unit is 0.1 mA. E.g. “READ?” might return #18<0x10><0x01><0x0><0x0><0x40><0x0><0x0><0x0> This indicates that 1 sample has been acquired for each port. The first port returned 0x10,0x01 which is 272 as a 16-bit number, corresponding to 27.2 mA. Port 2 and 4 are 0 mA, while port 3 measures 6.4 mA.
READ:ASC?	Returns the last acquired samples as comma-separated numbers in bundles of 4 samples, and clears the sample buffer. E.g. “READ:ASC?” could return: 272,0,64,0,217,0,67,0 Corresponding to two acquired samples. 27.2, 0.0, 6.4, 0.0 mA for the first sample. 21.7, 0.0, 6.7, 0.0 mA for the second sample.

10 Local application servers

The testbed is connected to the Internet, so any external server is accessible. However, in order to provide a more controlled environment Triangle has deployed the Asterisk VoIP server [26], a well-known open source solution, and a simplified DASH (Dynamic Adaptive Streaming over HTTP) server [29]. These servers have been also used to validate the initial deployment of testbed through the testing of CSipSimple VoIP client application [27] and Exoplayer DASH client [28], both for Android devices.

Asterisk [26] is an open source framework that provides an IP PBX (Private Branch Exchange) software solution. In order to automate the establishment of the calls, the Asterisk server is configured to provide a callback service, so that this service reproduces a 30-second recording each time a call is received in a preconfigured VoIP extension. Records have been extracted from audio samples provided in the recommendation P.501 [20] to speech quality evaluation on telephone networks.

The video streaming server is a NodeJS for DASH streaming using GPAC [29]. GPAC is an Open Source multimedia framework which offers support for DASH Streaming [21].

11 Preliminary testbed test experiment

This section describes a preliminary experiment with several of the components deployed in the TRIANGLE testbed. This experiment focuses on testing a well-known reference App, YouTube, on one device. It did not make use of all the integration and orchestration capabilities added to the first release of the testbed, but was instrumental in detecting requirements for those components.

11.1 YouTube Testing

The components provided by some partners have already been integrated into a first version of the TRIANGLE testbed. As an exercise the testbed has been used to evaluate the performance of a device when running a reference App. The test campaign included the test cases needed to evaluate the power consumption of one of the Triangle testbed reference applications: YouTube. The testing was run emulating a single user playing a YouTube video on the mobile device, but under different radio channel conditions (scenarios). An agent that drives the YouTube API was installed on the device under test, a Samsung Galaxy S4, connected to the UXM LTE network emulator via a coaxial cable and from there to the YouTube server via internet. The performance tool from AT4 was controlled remotely to run the agent and record KPIs using the algorithms proposed in [16].

The video used has a length of 5 minutes and resolutions ranging from 1080p to tiny. Power consumption is measured using the DC power analyser which is directly connected to the power terminals of the UE. The beginning of the YouTube session is synchronized with the power consumption capture. Once the power consumption measurement is initiated it lasts 120 seconds, independently of the duration of the playback of the video. To determine the total consumption, the power measurements have to be extrapolated considering the total playback time.

The scheduler used during the testing only transmits LTE PDUs to the mobile device (in the downlink) when there is IP data to carry. The UE transmission power is controlled automatically to be kept at 0 dBm in all the scenarios for better analysis of the impact of the DL configurations.

In the frequency domain, the full channel bandwidth has been assigned for downlink transmission in the UXM. This translates into 100 Physical Resource Blocks (PRBs) in a 20 MHz cell.

Different levels of Additive White Gaussian Noise (AWGN) have been also added when required to model interference from neighbour cells transmitting at the same frequency. The Signal to Noise Ratio (SNR) is determined by the relation between the power density of the transmitting cell and the interfering noise.

The downlink transmissions follow an open-loop transmit diversity (TM3) MIMO scheme, and the modulation and coding schemes are dynamically adapted to the channel quality and rank reported periodically by the UE.

Channel models emulate the effects of the environment on the radio signal: fading, delay spread and Doppler shift.

Two different scenarios have been considered: A first scenario which emulates the behaviour of a channel while the mobile user is walking in a city (EPA5, 20 dB SNR) and a second scenario which represents a mobile user travelling by bus in a city (EVA5, 10 dB SNR).

The configuration of the cellular network includes 4G LTE cells with two transmitting antennas, resulting in a 2x2 MIMO matrix as all LTE devices have minimum two receiving antennas. The UE reports the observed radio channel quality (CQI) and the rank (RI), the later representing the number of independent data flows that can be transmitted on the channel.

11.2 Results

The YouTube traffic has a bursty nature. Figure 38 depicts the downlink (DL) traffic captured by TestelDroid during one of the test executions.

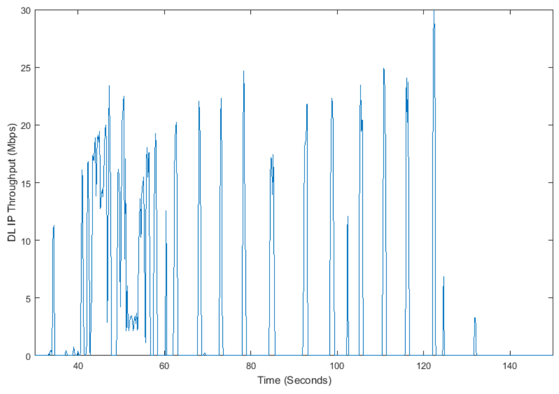


Figure 38 DL IP throughput during a YouTube session

At the beginning of the session, a larger throughput is observed as the application seeks to gather enough data before starting the playback.

Figure 39 displays the CQI and RI histogram recorded during the test cases execution under the pedestrian scenario. CQI is the mechanism used in LTE by the UE to report what modulation and coding scheme the eNodeB should use to achieve a nominal block error rate in the order of 10% at the link level for initial transmissions as stated in 3GPP TS36213 [22]. In terms of RI, both rank 1 (CW0) and rank 2 (CW1) were reported by the UE during the test, although it can be observed that rank 1 was the most frequently reported value. The transmission will most of the time just use one of the flows (corresponding to RI = 1), which implies lower capacity but less interference on the second flow.

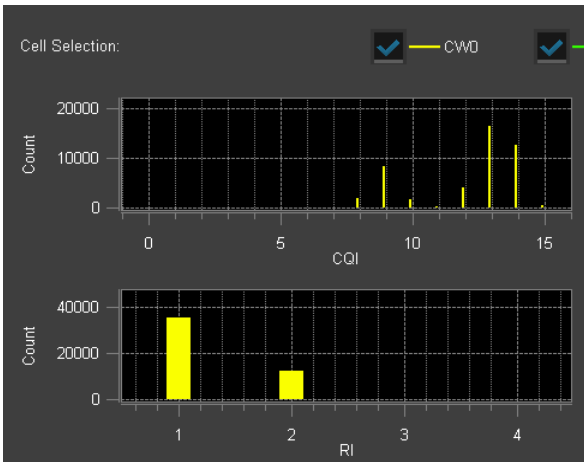


Figure 39 Channel quality and rank reported information in a pedestrian scenario

Table 35 for the pedestrian scenario and Table 36 for the vehicular scenario display the number of transmissions successfully acknowledged (ACK) in the first transmission (1st) and, when they occurred, in the retransmissions (2nd, 3rd, 4th). Table 35 differentiates the acknowledged messages for rank 1 (CW0) and rank 2 (CW1).

Table 35 DL retransmissions in a pedestrian environment

	% ACK CW0	%NACK CW0	% ACK CW1	% NACK CW1
1st	87.02	12.97	91.64	8.36
2nd	96.00	1.77	78.72	6.91
3rd	13.95	32.56	10.00	35.00
4th	5.41	43.24	5.56	44.44

In the pedestrian scenario the first transmission has a high success rate (87,76%), the second transmission (i.e., the first retransmission), also has a high success rate (96%), but in the third and fourth transmissions only a small fraction of the data manage to get through. This is a symptom of bad channel conditions, where the upper layers (RLC and TCP) shall be in charge of ensuring packet delivery without loss of information by increasing transmission at their corresponding levels.

In the vehicular scenario, the selected signal to noise ratio is 10 dB lower than in the pedestrian. This has been chosen to represent the lower strength to be expected in the vehicle. It also has a higher delay spread which is aggressive in terms of multipath. Both factors result in worse propagation conditions being reported by the mobile device under test. As shown in, Figure 40, all the recorded rank indications include requests for single layer (non-MIMO) transmissions (rank 1). Additionally, the reported quality (CQI) is lower than in the pedestrian case.

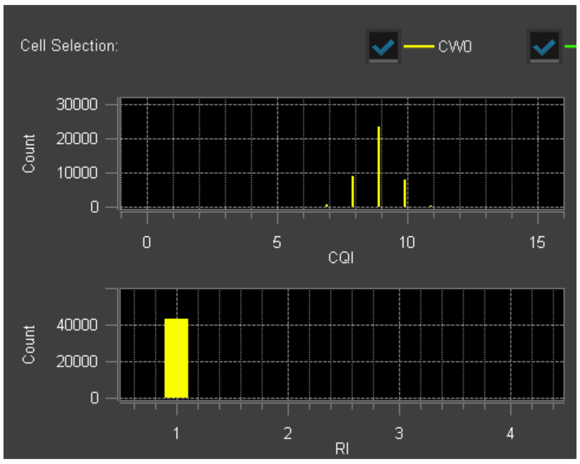


Figure 40 Channel quality and rank reported information in a vehicular scenario

Table 36 DL retransmissions in a vehicular environment

	% ACK CW0	% NACK CW1
1st	90.83	9.17
2nd	100	0
3rd	0	0
4th	0	0

The different radio channel conditions impact the final QoE of the YouTube sessions. Table 37 provides a summary of KPIs collected by the Performance Tool and a QoE score according to the method described in [16]. With only small differences in the initial buffering time, low number of re-bufferings (0) and zero re-buffering time (0) the resulting MOS score is high

(>3.5). The video quality, not considered in the algorithm used to evaluate the MOS, but also remains constant over time (see Figure 41).

Table 37 YouTube KPIs in single user scenarios

	EPA	**EVA**
Initial buffering	3.551	2.908
Playback Size (MB)	46.067	45.979
Playback Time (s)	160.638	160.982
MOS	3.89	3.89
% time in 1080p	100	100
	EPA	EVA

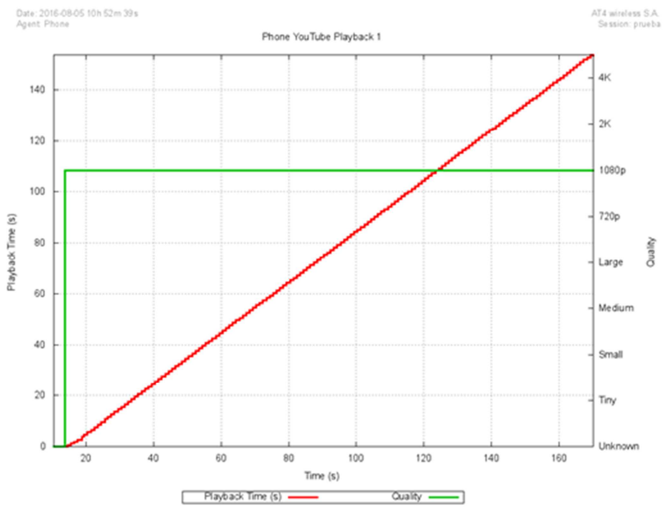


Figure 41 Playback time and video quality versus time

In terms of power consumption, only moderate differences have been found. Figure 42 displays the averaged power consumption over time, with a sampling period of 100 ms. Only during the 10 initial seconds, averaging during 5 different sessions per test condition, the power consumption has been observed to be 11% higher in the vehicular scenario than in the pedestrian scenario (as shown in Figure 10). After the initial buffering and processing peak, the power is observed to be similar in both scenarios.

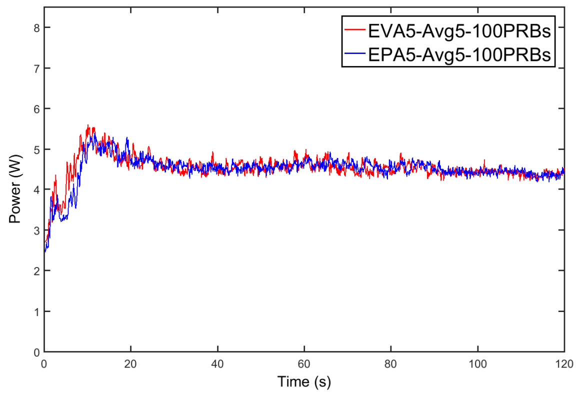


Figure 42 Power consumption over time

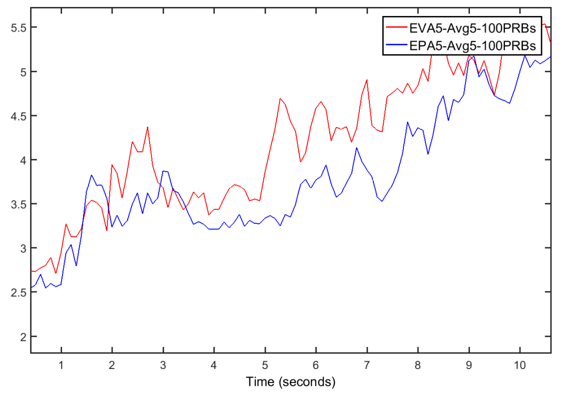


Figure 43 Power consumption during initial buffering in single user scenario

To provide more realistic insight, the tests have been repeated allocating only 8 PRBs. This resembles a dense environment where the resources of the cell are shared with other users during the session. Under these more constrained conditions, remarkable differences between both scenarios can now be observed.

In the pedestrian scenario, the available capacity does not allow to keep the full 1080p resolution, and YouTube automatically changes the video quality to 720p (Figure 44). In the vehicular scenario, where the radio channel conditions are tougher, YouTube service reacts dropping the video quality down into the large format early in the playback (Figure 45).

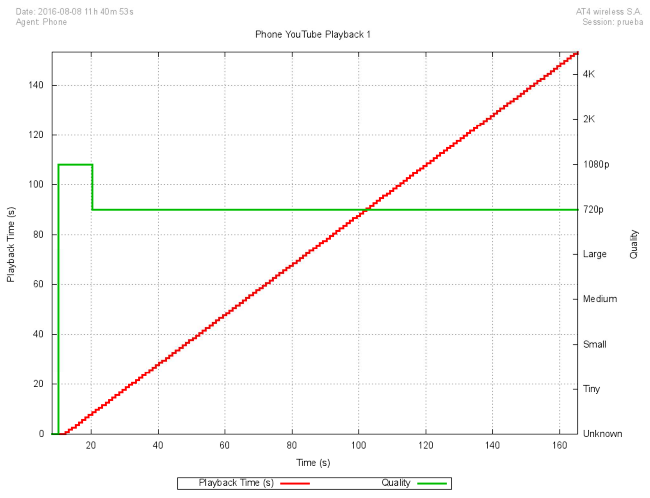


Figure 44 Playback time and video quality versus time in a dense pedestrian scenario

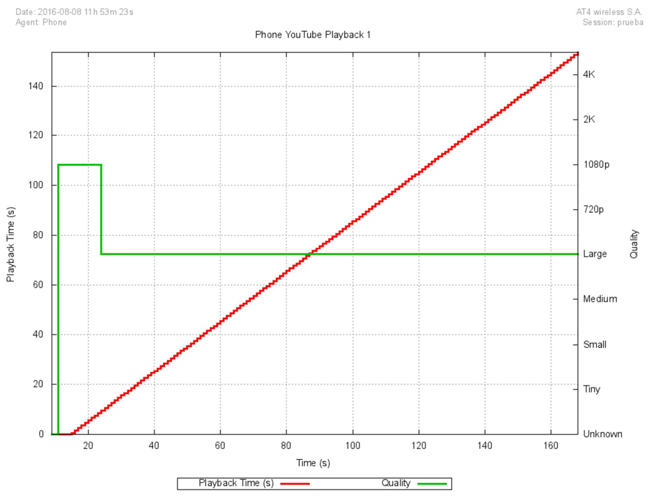


Figure 45 Playblack time and video quality versus time in a dense vehicular scenario

Figure 46 shows the power consumption over time. An increment in the power in the pedestrian scenario can be observed. This is caused by the data being fragmented in a larger number of segments when transmitted over the air because of the smaller amount of allocated PRBs. It must be noted that the mobile device has to process and transmit feedback in fewer occasions in the vehicular scenario because the video is transmitted with lower resolution. Normalizing the power per MB of video the impact becomes clearer. The large resolution requires 17 MB to be transmitted, whereas the 720p video generates only 29 MB of data. Thus, in the pedestrian scenario, the video can be reproduced with the double amount of data at the expense of similar power consumption.

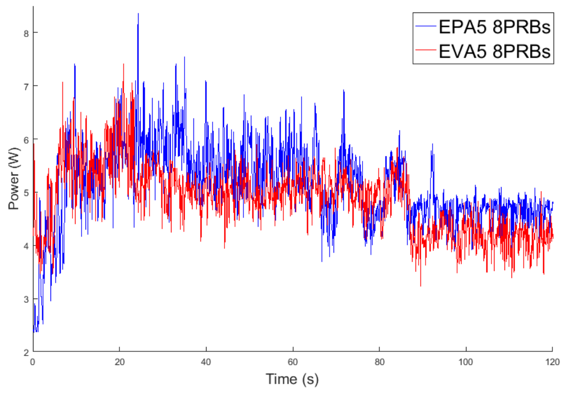


Figure 46 Power consumption in dense scenarios

Table 38 shows the relevant KPIs and how the MOS is slightly reduced. Because the MOS score algorithm in [16] does not consider the video quality the project has identified an area of research to establish relevant KPIs in dynamic video quality adjustment conditions.

Table 38 YouTube KPIs in dense scenarios

	EPA5	**EVA5**
Initial buffering (s)	3.325	5.416
Playback Size (MB)	29.425	17.172
Playback time (s)	157.639	159.733
MOS	3.89	3.81
% time in		
1080p	5,23	5.84
720p	94.77	0
Large	0	94.16

11.3 Importance of KPIs alignment

The synchronization of the different measurements is very important to enable further analysis and to infer dependencies between different factors. Having synchronized the power monitoring with the application sessions has allowed TRIANGLE to easily characterize the average behaviour of multiple YouTube sessions. Even finer details can be observed within individual sessions, as the instantaneous power consumption is highly correlated with data traffic burstiness.

12 IoT device proof of concept

The BlueEye wearable system will be treated as an alpha customer of the Triangle testbed for IoT devices. The quick feedback loop enabled by the use of this system will allow the continuous evaluation of the solutions proposed in the project for dealing with the complex IoT domain.

12.1 Overview of BlueEye

BlueEye is a wearable video system based on a glass-mounted camera including audio, which enables live interactive point of view video to be streamed to a command center, hospital emergency department or hot desk to provide real time interactive support.

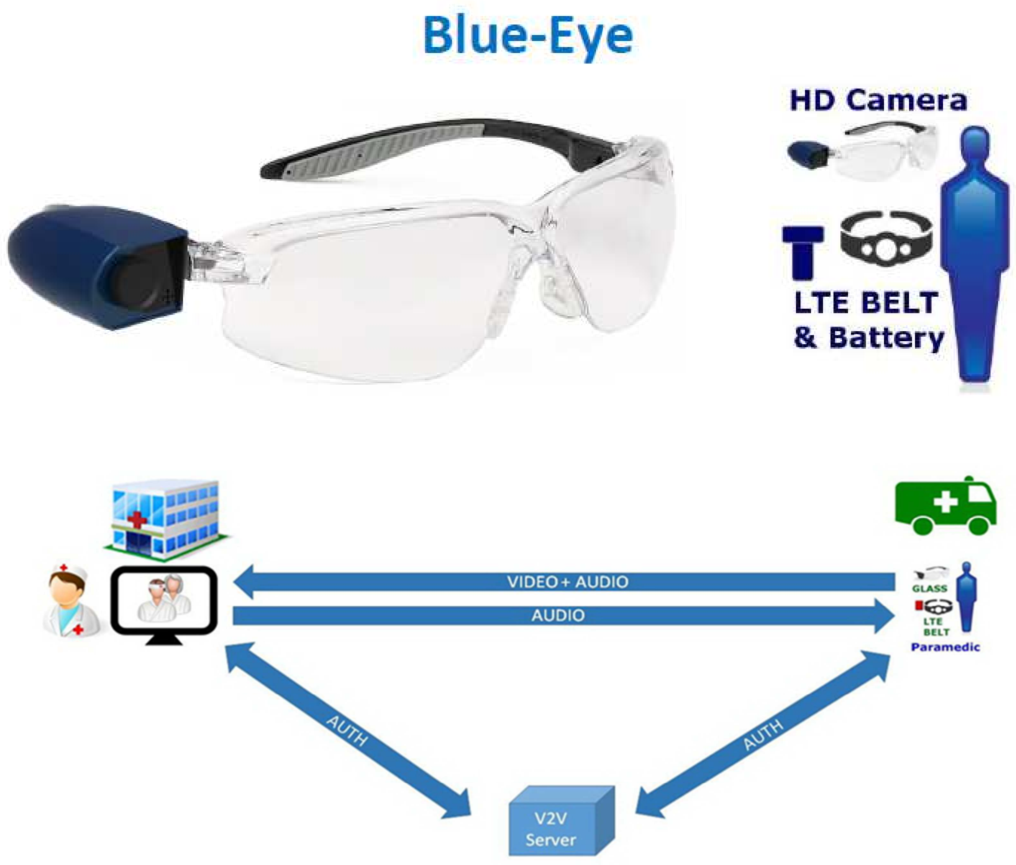


Figure 47 BlueEye system

12.2 Technical specification

This section provides the technical specification of the BlueEye system.

Table 39 BlueEye specifications

Specifications

BlueEye User Equipment Data	
Camera	• Safety Glasses Mounted • 1/2.7-inch sensor size • 1080p/720p HD Colour CMOS Image • Diagonal 110 +/5-degree wide angle • H.264 Codec • Internal Microphone Up to 1080p @ 30 fps (depending on uplink bandwidth
Processor	• Low Power Intel® Edison • Dual core Wearable Processor
LTE & Wireless modem	• 3GPP Release Baseline 9 • LTE FDD Category 3 • LTE Bands 3, 7, 20 • American LTE Bands on Request • 802.11abgn Dual SIM
Battery	• Replaceable Lithium Ion • 49 W/h • Nominal operating voltage is 7.2V. • Current limit 3.0 A. • Active mode current consumption <745 µA • Standby mode current consumption <200 µA • Shut-down mode current consumption <1 µA. • Voltage resolution 1 mV. Current resolution 1 mA.
Video & Audio Connectivity	• Live Streaming to Hot Desk or Hub • Simplex Video • Duplex Audio Encrypted Video and Audio
Indicators	• System activated, • Camera connected Connection with Hospital centre established.
System Data	
Hub Application	• Customisable Application for Hot Desk • Video Image Selectable from each field camera. • Image Snapshot • Deployable at: o Hospital Hot Desk o Emergency Department o Police Command Centre o Service Centre Hub Select from Multiple UE Video Streams
Measurement	• UE Based LTE Capacity Probe

	• In Line Video Throughput Probe Dashboard with pre-call measurements, in call measurements and location information (option on request)
Options upon request	• GPS using GPS, QZSS, GLONASS • Location Map for location stamp for evidence gathering (Depend on GPS Option) • Traffic Prioritisation on LTE in partnership with 4G provider • Video Repository Integrate to video repository for evidence, clinical data and other requirements for later viewing.

12.3 System connections

In Figure 48 the location of the different components of the system and their connections along the main board are described.

During testing the antennas connection will be realized following the scheme showed in Figure 51. Battery connections are detailed in Figure 49.

Figure 52 depicts both faces of the Panel board where the main component are the Led indicators and the buttons to turn on the System and the video camera, together to the the DC and battery connections.

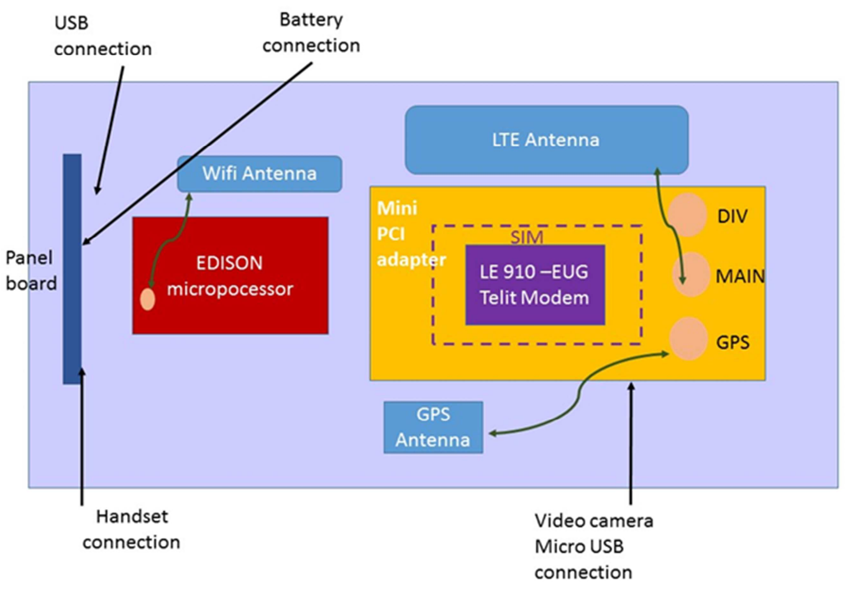


Figure 48 BlueEye internal diagram

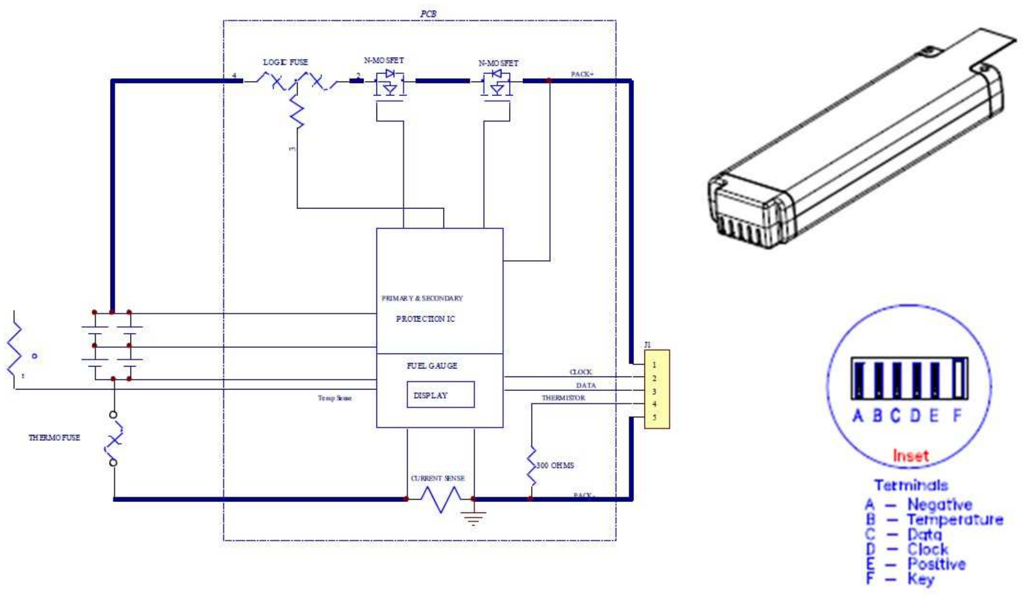


Figure 49 DC and battery connections

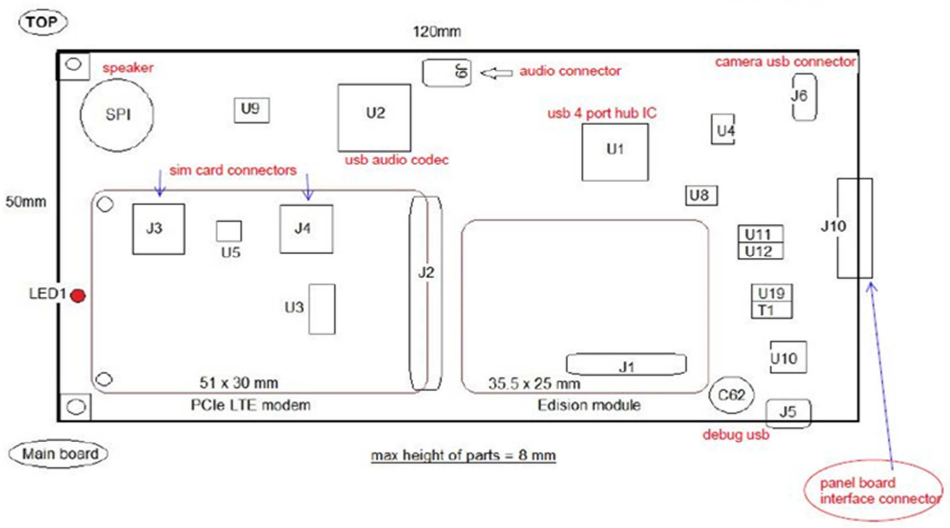


Figure 50 Main board

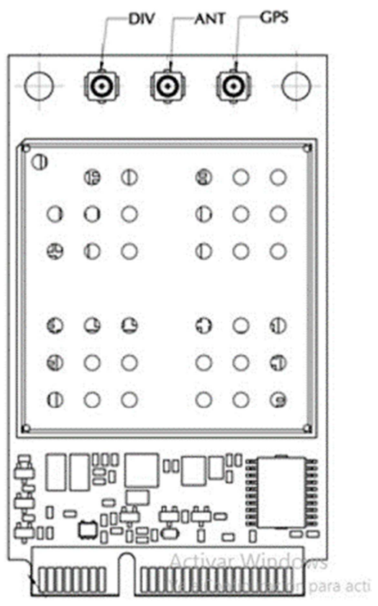


Figure 51 Telit antenna connectors

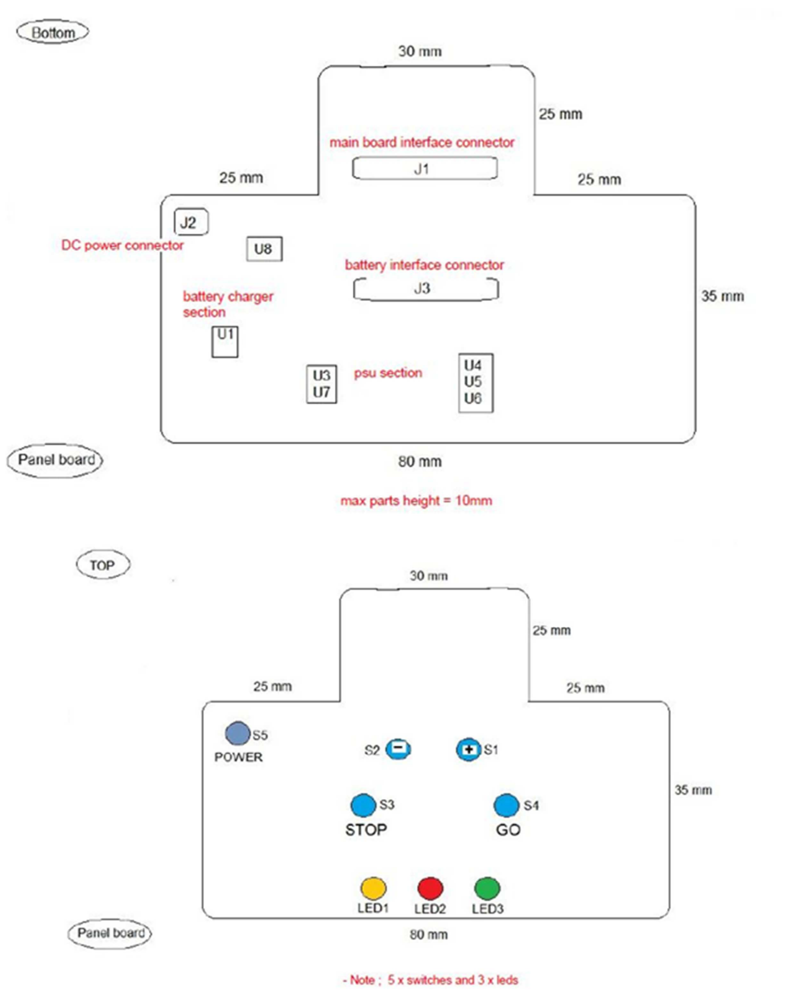


Figure 52 Panel board

12.4 Control and configuration instructions

The process to control the device from the testing system requires to know how to configure and initiate the system.

The main configuration and control instructions are detailed bellow.

Table 40 BlueEye commands: switch system on

	Commands

(from Edison microprocessor)
Update the date
	`# date --set 2016-09-22`
	`# date --set 11:00:00`

Confirm that all components are recognized	`# lsusb`
In order to communicate with the modem ii is necessary to establish a ppp0 connection by Wvdial. *The Wvdial script is located in:*	`# wvdial <name>`

/etc./wvdial.conf *Call wvdial:*	(Now the modem is connected to LTE network and can connect to the Hospital Centre. Both, BlueEye and Hospital centre are connected to a private VPN)

Table 41 BlueEye commands: press button to turn on video camera

	Commands

Executing this script "go", Edison understands that the Video camera button is pressed.	`#cd/etc./openvpn/`
	
	`/etc./openvpn# service openvpn start blueeye-client`
	
	`# cd /home/edison/`
	
	`/home/edison# ./go`

12.5 BlueEye in Triangle testbed

The information provided in previous sections has been used to connectorize the RF antennas of the IoT system and connect the power consumption probes as shown in Figure 53.

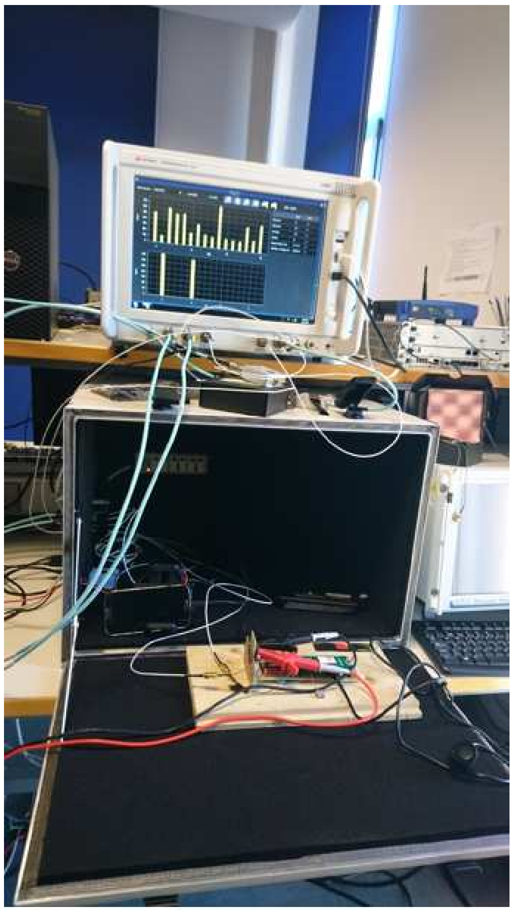


Figure 53 BlueEye connectorization

13 Future extensions

This section provides a brief description of planned extensions.

13.1 Wi-Fi access points

The AT4 wireless Performance Tool (section 5.2) AT4-Agents (Windows version) will include the features to automate many WLAN Access Points simultaneously in the next release..

The goal of this feature is to automate roaming scenarios, channel sweep, WLAN to Cellular transitions, and 5G dual connectivity.

This feature requires that the WLAN AP provides a Telnet control interface. The AT4 wireless Performance Tool will use that interface to send commands and automate the Wi-Fi Access Point.

The following actions wil be automated in the next release of the Triangle testbed:

1. Turn on/off radio
2. Set channel and bandwidth
3. Set transmission power

The WLAN AP automation scanning architecture is depicted in the next figure.

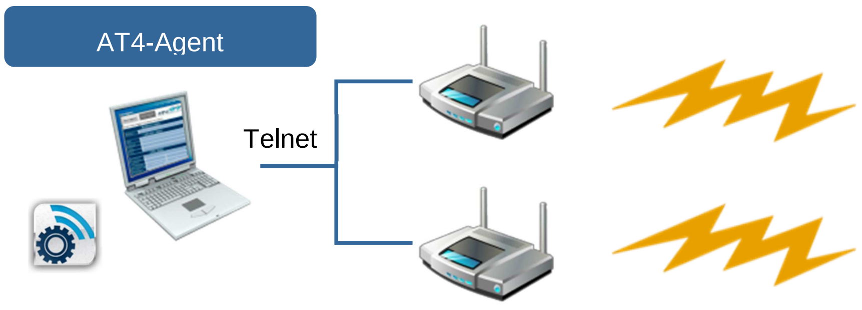


Figure 1 - Wi-Fi Access Point Automation

The Appendix 1 provided the details of the XML file which specifies the specific command for a given Wi-Fi Access Point model.

13.2 GPS

In order to support the testing of virtual reality applications Triangle is evaluating how to include GPS simulation feature in the testbed. The following solutions have been identified:

HW:

- HackRF http://www.seattletechnicalbooks.com/hackrf
- BladeRF https://www.nuand.com/blog/product/bladerf-x40/
- URSP https://www.ettus.com/product/category/USRP-Bus-Series

Software:

- Software-Defined GPS Signal Simulator (MIT license, free) https://github.com/osqzss/gps-sdr-sim

13.3 Small cells

The University of Malaga is currently negotiating with several vendors the acquisition of new small cells supporting releases 10/12. In the meantime, there are already some small cells available for the project.

Pixie Small Cells, which were provided by Athena Wireless (now part of Google Fiber). These cells work on band 7 and have a maximum transmit power of 2W. The configuration can be done via a CLI interface and theoretically also using a TR.069 OAM interface.

Alcatel Lucent (now part of Nokia) pico cell prototype; this cell was donated by Alcatel Lucent and offers a very complex configuration interface very similar to the one present in most macro cells used by Tier 1 operators. The base station supports dedicated bearer traffic differentiation as well as many other useful features.

13.4 Model based testing

To provide a complementary approach to app testing, the TRIANGLE testbed will offer a model-based solution for app testing. In the proposed approach [9], app developers will describe the app behaviour using models. These models define the expected behaviour from the point of view of the end-user of an app, i.e. how to interact with the app to accomplish certain tasks, and how the app reacts to that interaction. This model will be explored to extract all possible interaction paths allowed by the model. While some behaviour fragments are deterministic, their aggregation in a model allows exploring combinations that were not foreseen initially.

Testing of some specific KPIs, e.g. time to login, require very specific sequences of user actions in order to provide a repeatable and measurable path. Model-based testing could be used as another source of app quality benchmarking, e.g. helping uncover app functional errors by increasing coverage. However, this approach can also be used to check extra functional properties, i.e. properties that are not related to app functionality, such as battery consumption [10], and could thus be used to complement existing KPIs.

14 References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015-2020 White Paper, February 2016

[2] The dummynet project: http://info.iet.unipi.it/~luigi/dummynet

[3] http://www.keysight.com/en/pd-1842303-pn-N6705B/dc-power-analyzer-modular-600-w-4-slots?pm=PL&nid=-35714.937221&cc=DK&lc=dan

[4] https://w3c.github.io/webdriver/webdriver-spec.html

[5] OML Measurement Library: https://oml.mytestbed.net/projects/oml/wiki/

[6] Quamotion WebDriver, Configuring Developer Profiles: http://docs.quamotion.mobi/en/latest/getting-started-developer-profile.html

[7] WebDriver: https://www.w3.org/TR/webdriver/

[8] Quamotion.Webdriver client: https://github.com/quamotion/Quamotion.WebDriver

[9] Xamarin Acquaint app: https://github.com/xamarinhq/app-acquaint

[10] Logcat Command-line Tool: https://developer.android.com/studio/command-line/logcat.html

[11] Ruby on Rails: http://rubyonrails.org/

[12] Bootstrap front-end framework: http://getbootstrap.com/

[13] Vagrant by HashiCorp: https://www.vagrantup.com/

[14] Ana Rosario Espada; María del Mar Gallardo; Alberto Salmerón y Pedro Merino, “Using model checking to generate test cases for Android applications”, XV Jornadas sobre Programación y Lenguajes (PROLE 2015)

[15] Ana Rosario Espada; María del Mar Gallardo; Alberto Salmerón y Pedro Merino, “Runtime verification of expected energy consumption in Smartphones”, 22nd International Symposium, SPIN 2015 (DOI: 10.1007/978-3-319-23404-5_10.)

[16] F. Lozano, et al., “Network Performance Testing System Integrating Models for Automatic QoE Evaluation of Popular Services: YouTube and Facebook”, Wireless Personal Communications, Springer US (DOI = 10.1007/s11277-015-2479-y).

[17] D2.1 Initial report on the testing scenarios, requirements and use cases, Appendix 2

[18] http://www.keysight.com/en/pd-2372474-pn-E7515A/uxm-wireless-test-set?cc=US&lc=eng

[19] http://www.flex-project.eu/

[20] ITU-T P.501: Test signals for use in telephonometry

[21] ISO/IEC 23009-1:2014 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats

[22] 3GPP TS 36.213 Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures

[23] Thierry Rakotoarivelo, Max Ott, Guillaume Jourjon, Ivan Seskar, "OMF: a control and management framework for networking testbeds", in ACM SIGOPS Operating Systems Review 43 (4), 54-59, Jan. 2010

[24] Olivier Mehani, Guillaume Jourjon, Thierry Rakotoarivelo, and Max Ott, "An instrumentation framework for the critical task of measurement collection in the future Internet," Computer Networks, vol. 63, pp. 68-83, Apr. 2014

[25] Android Debug Bridge: https://developer.android.com/studio/command-line/adb.html

[26] Asterisk custom communications: http://www.asterisk.org/

[27] CSipSimple VoIP client: https://code.google.com/archive/p/csipsimple/

[28] ExoPlayer: https://google.github.io/ExoPlayer/
[29] GPAC Multimedia Open Source Project: https://gpac.wp.mines-telecom.fr/

15 Appendix 1: AT4 wireless Performance Tool Capabilities

This appendix summarizes the list of measurement capabilities provided by the AT4 wireless Performance Tool which have been integrated into release 1 of the Triangle testbed.

15.1 QoS Measurements

Table 42 Performance Tool QoS Measurement Capabilities

QoS Measurements	
KPI Resolution	One KPI record per Layer 7 SDU
Throughput	Over Time, Average, CDF (Cumulative Distributed Function), PDF (Probability Distributed Function)
One-way Delay	Over Time, Average, CDF, PDF, Percentile (t)
One-way Delay Variation	Over Time, Average, CDF, PDF, Percentile (t)
One-way Packet Loss Rate	Over Time, Average, Consecutive Loss Packets
One-way Packet Loss Distribution	RTT, Number of retransmissions, Duplicated ACK, Windows Size
Others	Inter-Departure-Time, Inter-Arrival-Time

15.2 API-Driven QoE Measurements

Table 43 Performance Tool QoE Measurement Capabilities

App	**QoE Measurement**
VoIP	Estimated MOS
*YouTube^TM^*	Playback Quality (t), Playback Time (t), Quality Usage CDF, Playback Size/Duration, Bufferings, MOS*
*Spotify^TM^*	Track Size, Playback Duration, Bufferings, MOS*
*Facebook^TM^*	Time to post text/pic/video
Web Browsing	Time to load web page
File Transfer	Time to transfer the file
Ping	RTT

* estimaded MOS

15.3 RF and System Measurements at UE

Table 44 Performance Tool RF and System Measurements

Type	**Measurements**
WLAN	RSSI, Noise Level, PHY Rate, Channel, SSID, Link Quality, Roaming
3G/GPRS	MNC, MMC, LAC, RSSI, BER, RAT, Cell-Id

CDMA/EVDO	System ID, Network ID, BSID, Cell-Id, RSSI, ECIO
LTE	Cell-Id, TAC, RSSI, RSRQ, RSRP, SNR
System	Data Connection In Use (WLAN/Cellular), Data Interface Usage (Mbit/s), Battery (%), CPU Usage (%), GPS coordinates

15.4 Built-in Traffic Generator

Table 45 Performance Tool Built-In Traffic Generator

Type	**Properties**
Protocols	TCP, UDP, IPv4, IPv4
SDU Size	Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson, Gamma, Weibull
SDU Inter-Departure Time	
Pre-defined Profiles	VoIP (MOS), Max. Performance, Online Gaming, RTP video, Live TV, Live Radio, PTT
Supported Patterns	Ramps, Bursts, Loops

16 Appendix 2: AT4 wireless Performance Tool Automation Interface (Release 1)

16.1 Interface Description

AT4-Controller is constituted by two separate processes: The graphical user interface (GUI) and the controller itself (`PerformanceTester.exe`).

The automation described in this appendix consists on replacing the AT4-Controller GUI process by an external Automation Suite as depicted in the figure below.

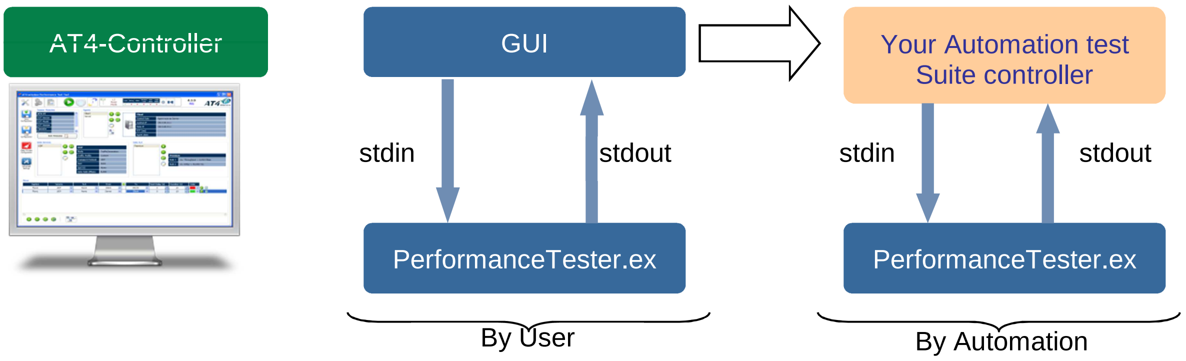


Figure 54 AT4 wireless Performance Automation Interface (rel'1)

Basically, the Automation Suite shall launch the process PerformanceTester.exe to execute a measurement:

```
C:\Program Files (x86)\AT4 wireless Performance Tool Controller\PerformanceTester.exe Input.xml
```

The process PerformanceTester.exe needs one argument called Input.xml which contains the required configuration to execute the measurement.

- `Input.xml` is a file generated by the GUI process right after “Play” button is clicked by the User in the regular way (i.e., commanded by User through GUI). This file contains all the configuration information (agents, data profiles, etc.).
- `Input.xml` is described in section 3.

The Automation Suite can use *stdin* (section 4) and *stdout* (section 5) streams to exchange information with the process `PerformanceTester.exe` while the measurement is running.

`PerformanceTester.exe` generates the results files of the test in a folder that can be learned by the automation suite from *stdout* messages.

16.2 Recommended automation methodology

The recommended automation methodology is to record into an XML file, the set of steps by first executing them manually.

To do so, for a given test case (Test Case A):

1. Open the GUI

2. Create the Configuration in the GUI
3. Click Play button (even if the test setup is not ready, the XML file is generated just after clicking Play button
4. Go to %TEMP% and catch "Input.xml".
5. Rename "Input.xml" to "Test_Case_A_input.xml"

For other test cases repeat steps 2 through 5. At the end a sort of library of reference input XML files should be available.

Accordingly, the Automation Suite (e.g., script) can be something like this:

```
//Input Parameters:
    •   A. Test_Case_X_input.xml for a given Test Case "X"
    •   B. Actual Agents Information
    •   C. Number of iterations

//Script Content
Actual_Input.xml=Modify(Test_Case_X_input.xml,B)
For i=0;i<C
     execute(PerformanceTester.exe Actual_Input.xml)
```

The goal of generating the reference XML files before executing any test case is to minimize the complexity of the "`Modify`" function in the Automation Suite.

Ideally, once this is done, the Automation Suite should just need to change AT4-Agents IP addresses from the XML file. The script should not need to modify more fields from the XML file.

16.3 Input XML file

This section describes the structure of `Input.xml` file.

All fields are mandatory.

The fields that may need to be modified in the Automation Suite have been highlighted in red.

```
<input>
  <CaptureTraffic>
    <main_path/> Working directory of the tool (1)
    <temp_path/> Temp folder of the system
    <session_folder/> Name of the project (1)
    <time_folder/> If set to "none", folder with data and time will be created. Otherwise the given folder will be used
    to store the results (folder must exist) (4)
    <repetition_folder>none</repetition_folder> Do not modify
    <PTversion>50007</PTversion> Do not modify
```

[4] Results are generated at <main_path><session_folder><time_folder>

```
    <pt_user>AT4 wireless</pt_user> Do not modify
    <generate_record_file>0</generate_record_file> GUI: Post Mortem Recovery (1:Yes, 0:No) Do not modify
    <record_file_interval>60</record_file_interval> GUI: Interval Between Backups. Do not modify
    <delete_record_file>1</delete_record_file> GUI: Delete Backup Files After Test. Do not modify
    <basic_mode/> GUI: Test Mode (0: Basic Mode, 1: Advanced Mode)
    <using_wpcap/> GUI: Test Mode (0: L7 Mode, 1: L3 Mode)
    <time_duration>87000</time_duration> Do not modify
    <packet_duration>50000000</packet_duration> Do not modify
    <waiting_client_agents/> GUI: Control Plane Connection Timeout
    <waiting_server_agents/> GUI: Control Plane Connection Timeout
    <packet_loss_th/> GUI: Loss Threshold
    <poisson_lambda/> GUI: Graph Resolution
    <thput_PDF_interval/> GUI: Thput PDF Width (kbit/s)
    <delay_PDF_interval/> GUI: OWD/IPDV PDF Width
    <jitter_PDF_interval/> GUI: OWD/IPDV PDF Width
    <peak_percentile/> GUI: Peak Definition
    <wifi_interval>1000</wifi_interval> Do not modify
    <cell_interval>1000</cell_interval> Do not modify
    <generate_XML>0</generate_XML> Do not modify
    <compensate_clock_skew>1</compensate_clock_skew> Do not modify
    <max_jitter_supported/> GUI: Maximum Jitter
    <target_layer/> GUI: L3 Target Layer
    <sessionrestoretime/> GUI: Reconnection Period
    <maxconnectattempts/> GUI: Max Reconnection Attempts
    <udpnatkeepaliveinterval/> GUI: UDP NAT KeepAlive Period
    <udpnatkeepalive/> GUI: UDP NAT KeepAlive During Traffic Flow
    <tcpconnecttimeout/> GUI: Data Plane Connection Timeout
    <udpconnecttimeout/> GUI: Data Plane Connection Timeout
    <resenddata/> GUI: Retransmission Period
    <pt_os/> Version of Windows
    <pt_cpus/> Number of CPUs
    <pt_parallelw2008/> Parallelization in control plane
    <remove_samples/> GUI: Remove samples from KPI
    <graph_resolution/> GUI: Graph Size
    <graph_extension/> GUI: Graph Extension
    <generate_vector_delay/> GUI: Save KPI at Packet Level
    <generate_vector_jitter/> GUI: Save KPI at Packet Level
    <generate_vector_loss/> GUI: Save KPI at Packet Level
    <generate_vector_packetsize/> GUI: Save KPI at Packet Level
    <generate_vector_idt/> GUI: Save KPI at Packet Level
    <generate_vector_txtime/> GUI: Save KPI at Packet Level
    <map_max_thr/> GUI: Map Tput Max Value
<number_of_MP/> Number of Agents participating in the test. Each of them represented in a <MP> structure
    <MP>
        <caption/> GUI: Caption
        <index/> Agent Index: starting at 0, must be incremental
        <mode>PASSIVE</mode> Control Plane Mode: Do not modify
        <identifier>Agent1</identifier> Client Mode ID: Do not modify
        <version>50007</version> Version of the Agent (5.00.07) Must match!
        <capture_data/> If the Agent must capture data in L3 (0:No, 1:Yes)
        <wifi_stats/> GUI: Capture WLAN Parameters
        <wifi_scanning/> GUI: Capture WLAN Neighbours Parameters
        <external_wifi/> GUI: Obtain WLAN Info Remotely From (-1: Unused, Other: MP index)
        <cell_stats/> GUI: Capture Cellular Parameters
        <external_cell/> GUI: Obtain Cellular Info Remotely From (-1: Unused, Other: MP index)
        <system_stats>1</system_stats> Capture System Stats: Do not modify
        <com_port>None</com_port> GUI: Cellular COM Port (Windows only) Do not modify
        <sim_pin>None</sim_pin> GUI: Cellular SIM PIN (Windows only) Do not modify
        <gps_stats/> GUI: Capture GPS Info
        <external_gps/> Obtain GPS Info Remotely From (-1: Unused, Other: MP index)
        <is_sniffer>no</is_sniffer> Do not modify
        <sniffer_mac_offset>0</sniffer_mac_offset> Do not modify
        <management_ip/> GUI: Control IP
        <management_port/> GUI: Control Port
        <data_ip/> GUI: Data IP
        <phone_number>None</phone_number> Do not modify
        <number_of_attenuator>0</number_of_attenuator> Do not modify
        <number_of_ap_automation>0</number_of_ap_automation> Do not modify
        <number_of_powersources>0</number_of_powersources> Do not modify
        <number_of_turntables>0</number_of_turntables> Do not modify
        <wifi_monitor>0</wifi_monitor> Do not modify
        <use_dsla>0</use_dsla> Do not modify
        <dev_control>0</dev_control> Do not modify
    </MP>
    <number_of_flows/> Number of Flows in the test. Each of them represented in a <flow> structure
    <flow>
        <caption/> GUI: Caption
        <index/> Flow Index: starting at 0, must be incremental
        <group/> GUI: Group
        <streams> GUI: TCP Streams. For UDP set to 1
        <generateTraffic/> 1: Data Traffic Generator, 0: Data Traffic Monitor (L3)
        <MP_DSCP/> DSCP filter for L3, -1 to ignore
        <MP_SPI>0</MP_SPI> Do not modify
        <MP_transport/> Transport protocol filter for L3 (UDP/TCP)
        <MP_src_port/> Source port filter for L3, 0 to ignore
        <MP_dst_port/> Destination port filter for L3, 0 to ignore
        <MP_pdu_size/> PDU size filter for L3, 0 to ignore
<number_of_MPs_participating/> Number of Agents participating in flow (could be >2 if using Via) for L3. Each of
them represented in a <MP_info> structure
        <MP_info>
<MP_index/> Index of the Agent participating (matching <MP><index>)
<source_ip_address/> Source IP address filter for L3 for this Agent
<destination_ip_address/> Destination IP address filter for L3 for this Agent
        </MP_info>
        <is_MOS/> Compute MOS stats for this flow (should be UDP flow)
        <jBuffer/> GUI: Jitter buffer
        <codec/> If MOS stats computed, which codec should apply
        <TG_configure_trigger>0</TG_configure_trigger> Do not modify
         <TG_configure_delay/> GUI: If “Configure before Delay”: 0, If “Configure after Delay”: Delay of the flow
        <TG_start_trigger>0</TG_start_trigger> Do not modify
        <TG_start_delay/> GUI: Delay of the flow (ms)
        <TG_end_trigger>0</TG_end_trigger> Do not modify
        <TG_end_delay/> GUI: Duration of the flow (ms)
        <TG_transport/> GUI: Transport Protocol
        <TG_max_datarate/> GUI: Max Performance
        <TG_data_port/> GUI: Transport Port
        <TG_traffic_mode/> "active": Receiver acts as Server, "passive": Receiver acts as Client
        <TG_ip_sender_data/> Local IP address of the Sender
        <TG_ip_sender_data_public/> Public IP address of the Sender
        <TG_ip_receiver_data/> Local IP address of the Receiver
        <TG_ip_receiver_data_public>Public IP address of the Receiver
        <TG_number_steps>Number of TG Steps. Each of them represented in a <TG_step> structure
        <TG_repeat_steps> GUI: 0: Normal Mode, 1: Loop Mode
        <TG_step>
           <step_duration>86400000</step_duration> Duration of the step in ms. If it's the last step, set to 86400000
           <idt_type>C</idt_type> GUI: Type of SDU Rate, C for Constant
           <pkts_per_s1>830</pkts_per_s1> GUI: Number of SDU/s
           <ps_type>c</ps_type> GUI: Type of Packet Size, c for Constant
           <pkts_size1>753</pkts_size1> GUI: SDU Size
        </TG_step>
    </flow>
    <number_of_voice_calls>0</number_of_voice_calls> Do not modify
   </CaptureTraffic>
</input>
```

16.4 Stdin Interface

Once `PerformanceTester.exe` is launched, it can be commanded with two different commands which must be sent to the process using the *stdin* interface.

- "`C\n`": Stops the test. It is equivalent to clicking the Stop button (when operated by the GUI). This command is ignored if it is not sent between the occurrence of the *stodout* messages: "Capturing Traffic Starts" and "Capturing Traffic Ends".
- "`AX\n`": Ignore Agent with index X. Agent with index X will be ignored for the rest of the test (will be considered disconnected). Take into account that the Agent will continue performing the actions if it was configured to, and has not been stopped. It is equivalent to clicking Force Stop button (when operated by the GUI).

16.5 Stdout Interface

Once `PerformanceTester.exe` is launched, the process sends log messages through the *stdout* interface.

Each output line includes date and time (2016-03-07 08:32:23 – <log>). This string (date and time) has been omitted from the examples below for simplicity. Some other log messages have been also removed for simplicity.

The log messages are also written in the Controller_log_PT.txt file.

The following general rules apply to the logs generated by `PerformanceTester.exe`:

- <mp><x> refers to the Agent defined in the `input.xml` with index "x"
- <flow><x> refers to the Flow defined in the `input.xml` with index "x"
- <flow><x><mp><y> refers to the Flow defined in the input.xml with index "x". "y" indicates the agent participating in the flow (0 refers to the sender, 1 to the receiver)

The messages logs that can be interesting to be parsed by the Automation Suite have been highlighted in red in the next subsections.

16.5.1 Start of test messages

Below is an example of "start of test" log messages.

```
<pm><info><general><time_path>2016-04-06 08h 42m 28s</time_path></general></info></pm>
<pm><info><general><path>C:\Users\Administrator\Documents\AT4 Performance Tool\CPU Performance\2016-04-06 08h 42m 28s</path></general></info></pm>
```

Important: This log message indicates the root folder where the results files will be stored. In this example: C:\Users\Administrator\Documents\AT4 Performance Tool\CPU Performance\2016-04-06 08h 42m 28s\

Below is an example log messages that show when the execution phase starts.

```
<pt><version><32 bit>5.0.7<
<pt><sessionID>1457335943<
ConnectionPhase
===============
<pm><info><mp><x>Connecting...</x></mp></info></pm>
<pm><info><mp><x>Connected</x></mp></info></pm>
<pm><info><general>ConnectionPhase successfully done!</general></info></pm>

ConfigurationPhase
==================
<pm><info><mp><x>Configuring...</x></mp></info></pm>
<pm><info><general>ConfigurationPhase successfully done!</general></info></pm>

SynchronizationPhase
====================
<pm><info><mp><x>Calibrating...<
<pm><info><mp><x><sync_prog>0 to 100 value</sync_prog></0></mp></info></pm>
<pm><info><mp><x>Sync done<
<pm><info><general>SynchronizationPhase successfully done!</general></info></pm>

TestExecutionPhase
==================
<pm><info><general>Capturing Traffic Starts...</general></info></pm>
```

At this point the test has started. During the execution of the test, and before the reception of the “end time” message, the test can be stopped writing “C\n” in the stdin of PerformanceTester.exe

16.5.2 Traffic Flows Messages

Below is an example of "traffic flows" log messages.

```
<tg><info><flow><x>Traffic Flow Configured received from MP0
<tg><info><flow><x>Traffic Flow Started received from MP0
<tg><info><flow><x>Traffic Flow Ended received from MP0
```

```
<pm><info><general>Capturing Traffic Ends!</general></info></pm>
<pm><info><mp><x>Stopping Generation...<
<pm><info><mp><x>Stopping...<

SynchronizationPhase
====================
<pm><info><mp><x>Calibrating...<
<pm><info><mp><x><s2ync_prog>0 to 100 value</s2ync_prog></0></mp></info></pm>
<pm><info><mp><x>Sync done<
<pm><info><general>SynchronizationPhase2 successfully done!</general></info></pm>

GetRecordsPhase
===============
<pm><info><mp><x>GettingRecords...<
…
<pm><info><general>GetRecordsPhase successfully done!</general></info></pm>

Start of Test: 1457335945s 375005us
End of Test:   1457335957s 285005us
RFC5835 Duration 11 s 910000 us, 12 intervals of 1000 ms<

<pm><info><general><devices><results>\Devices\</results></devices></general></info></pm>

<pm><info><general><mp><x><agent_path>\AgentCaption\<
```

\AgentCaption\ stands for the relative folder where results for Agent (e.g., RSSI capture, battery, etc.), with index “x”, are being stored. Folder name matches Agent caption given in input.xml. In this example: C:\Users\Administrator\Documents\AT4 Performance Tool\CPU Performance\2016-04-06 08h 42m 28s\Agent Caption\

```
<pm><info><mp><x>Computing System results for Agent2<
<pm><info><mp><x><system><first_battery>10<
<pm><info><mp><x><system><last_battery>10<
<pm><info><mp><x><system><battery_duration>-1<
<pm><info><mp><x><system><avg_cpu>36.69<
<pm><info><mp><x><system><max_cpu>51<
<pm><info><general><mp><x>Total computation time: 0 s<
```

Example of output for UDP

Outputs for flows depends on the Mode (L3/L7, Basic/Advanced)

```
<pm><info><flow><x><mp><0><data_rate>5.000<
<pm><info><flow><x><mp><0><max_dr>5.000<
<pm><info><flow><x><mp><0><peak_dr>5.000<
<pm><info><flow><x><mp><1><delay_av>1.707<
<pm><info><flow><x><mp><1><delay_med>0.561<
<pm><info><flow><x><mp><1><delay_min>0.409<
<pm><info><flow><x><mp><1><loss>0.000<
<pm><info><flow><x><mp><1><period>0<
<pm><info><flow><x><mp><1><jitter>0.475<
<pm><info><flow><x><mp><1><data_rate>5.002<
<pm><info><flow><x><mp><1><max_dr>5.018<
```

```
<pm><info><flow><x><mp><1><peak_dr>5.018<
<pm><info><flow><x><results>\Flow 0 (Agent1~Agent2@UDP-5M)<
```

The path in the last line is the relative folder where results for Flow "x" will be stored. Folder name matches Flow caption given in input.xml. In this example: C:\Users\Administrator\Documents\AT4 Performance Tool\CPU Performance\2016-04-06 08h 42m 28s\ Flow 0 (Agent1~Agent2@UDP-5M\

-Example of output for TCP

Outputs for flows depends on the Mode (L3/L7, Basic/Advanced)

```
<pm><info><flow><x><mp><1><throughput>5.000<
<pm><info><flow><x><mp><1><max_throughput>4.999<
<pm><info><flow><x><mp><1><peak_throughput>4.999<
<pm><info><flow><x><mp><1><delay_av>0.021<
<pm><info><flow><x><mp><1><jitter>0.009<
<pm><info><flow><x><mp><1><data_txmed>6.275<
```

The average KPI results from flow <x> are highlighted above.

```
<pm><info><flow><x><results>\Flow 0 (Agent1~Agent2@TCP-5M)<
```

The path in the last line is the relative folder where results for Flow "x" will be stored. Folder name matches Flow caption given in input.xml. In this example: C:\Users\Administrator\Documents\AT4 Performance Tool\CPU Performance\2016-04-06 08h 42m 28s\ Flow 0 (Agent1~Agent2@TCP-5M\

```
<pm><info><flow><x>Generating TXT Data ...<
<pm><info><flow><x>TXT Data generation time: 0 s<
<pm><info><flow><x>Generating Graphs ...<
<pm><info><flow><x>Graphs generation time: 0 s<

<pm><info><flow><x><mp><0><data_txrx>6.245<
<pm><info><flow><x><mp><1><data_txrx>6.245<
<pm><info><flow><x><mp><0><time_txrx>9.993<
<pm><info><flow><x><mp><1><time_txrx>9.988<
<pm><info><flow><x>Total computation time: 1 s<

<pm><info><group><1><average>5.002<
<pm><info><group><1><peak>5.018<
<pm><info><general><groups>Generating Groups Graphs...</groups></general></info></pm>
<pm><info><group><1><path>\Groups\Group 1\<

<pm><info><group><1><all_flows>;0;
<pm><info><general>Graphs plotting OK</general></info></pm>
```

This indicates that the flows have ended correctly (results can be read from their respective folders). If the process ends and this line does not appear, an error has occurred.

Device Automation Tasks Messages
Example of output for Web Browsing

```
<pm><info><mp><x><devcontrol><web><avg_setup>0.136<
<pm><info><mp><x><devcontrol><web><avg_session>1.484<
<pm><info><mp><x><devcontrol><web><avg_data_rate>1.390<
<pm><info><mp><x><devcontrol><web><success>100.00<
<pm><info><mp><x><devcontrol><web><failed>0.00<
<pm><info><mp><x><devcontrol><web><dropped>0.00<
<pm><info><mp><x><devcontrol><web><total>1<
```

Example of output for Youtube

```
<pm><info><mp><x><devcontrol><youtube><init_buffer>1.197<
<pm><info><mp><x><devcontrol><youtube><n_rebuf>0.000<
<pm><info><mp><x><devcontrol><youtube><rebuf_index>0.000<
<pm><info><mp><x><devcontrol><youtube><max_rebuf>0.000<
<pm><info><mp><x><devcontrol><youtube><mos>3.897<
<pm><info><mp><x><devcontrol><youtube><playback>7.194<
<pm><info><mp><x><devcontrol><youtube><total_rebuf_time>0.000<
<pm><info><mp><x><devcontrol><youtube><success>100.00<
<pm><info><mp><x><devcontrol><youtube><failed>0.00<
<pm><info><mp><x><devcontrol><youtube><total>1<
```

16.5.3 End of test Messages

Below are examples of "end of test" log messages.

```
StatsCalculationPhase
=====================
<pm><info><flow><x>KPIs computation starts
<pm><info><flow><x>KPIs computation time: 0 s<

<pm><info><general><end_time>2016-03-07 08h 32m 41s</end_time></general></info></pm> Test has ended
```

This indicates the process `PerformanceTester.exe` is about to finish.

17 Appendix 3: VPS Engine Control API

17.1 Requests

17.1.1 Trigger Request

Trigger request starts a new session of an available service between two provided IPs. All services are started as unidirectional, with an optional argument to make it bidirectional. Used for initiating a service between two IPs.

Trigger request returns a dataset containing the response code and the session ID of the running service in case of a successful trigger.

The Trigger request message format is shown in the table below.

Table 46 VPS Engine: Trigger request

Field	*Type*	*Description*
key	String	API Key
type	1	Request type, must be 1
source	String	IPv4/6 of the source point
destination	String	IPv4/6 of the destination point
service	Integer	ID of the service to trigger
bidirectional	Boolean	OPTIONAL: set to “true” for bidirectional service, omitted or set to “false” for unidirectional service.

TRIGGER Request Example

```
{"key":"ECE335024E3E466CA98BF5014D5C7D86", "type":1, "source": "200.20.10.32",
"destination": "193.22.33.55", "service": 16}
```

The format and an example of the data set response to a Trigger request message are shown below.

Table 47 VPS Engine: Trigger response

Field	*Type*	*Description*
type	Integer	Is 1 if successful or 0 in case of a generic error
code	Integer	Assumes a value from the Response Codes table
session	String	Alphanumeric string of variable length that uniquely identifies a service session, will be empty in case of error

TRIGGER Response Example

```
{"type":1, "code":0,
"session":"25172.16.14.10172.16.3.1300.40368552307899551332756705891"}
```

17.1.2 Stop Request

Stop request stops the running service session provided by ID. It returns a dataset containing the response code.

The Stop Request message format is shown in the table below.

Table 48 VPS Engine: stop request

Field	***Type***	***Description***
key	String	API Key
type	2	Request type, must be 2
session	String	Alphanumeric string that identifies the service session to stop

STOP Request Example

```
{"key":"ECE335024E3E466CA98BF5014D5C7D86", "type":2, "session":
"25172.16.14.10172.16.3.1300.403685523078995513327567O5891"}
```

The format and an example of the data set response to a Stop Request message are shown below.

Table 49 VPS Engine: Stop response

Field	***Type***	***Description***
type	Integer	Is 2 if successful or 0 in case of a generic error
code	Integer	Assumes a value from the Response Codes table

STOP Response Example

```
{"type":2, "code":0}
```

17.1.3 Modify Request

Modify Request modifies a running session provided by ID, changing the base service used. Itreturns a dataset containing the response code.

The Modify Request message format is shown in the table below.

Table 50 VPS Engine: Modify request

Field	***Type***	***Description***
key	String	API Key
type	3	Request type, must be 3
session	String	Alphanumeric string that identifies the service session to stop
service	Integer	ID of the service to modify the session to

MODIFY Request Example

```
{"key":"ECE335024E3E466CA98BF5014D5C7D86", "type":3,
"session":"25172.16.14.10172.16.3.1300.40368552307899551332756705891", "service": 14}
```

The format and an example of the data set response to a Modify Request message are shown below.

Table 51 VPS Engine: Modify response

Field	*Type*	*Description*
type	Integer	Is 3 if successful or 0 in case of a generic error
code	Integer	Assumes a value from the Response Codes table

MODIFY Response Example

```
{"type":3, "code":0}
```

17.1.4 List Request

List Request generates a list of services available for Trigger and Modify Requests. It returns a dataset containing the service list.

The List Request message format is shown in the table below.

Service Format:

Table 52 VPS Engine: List request, service format

Field	*Type*	*Description*
id	Integer	Service ID
name	String	Service Name
type	String	Type of Service handling inside the operator (Ex: Expedited Forwarding)
bandwidth	Integer	Service Speed in kbps

Table 53 VPS Engine: List request, message format

Field	*Type*	*Description*
key	String	API Key
type	4	Request type, must be 4

LIST Request Example

```
{"key":"ECE335024E3E466CA98BF5014D5C7D86", "type":4}
```

The format and an example of the data set response to a List Request message are shown below.

Table 54 VPS Engine: List response

Field	*Type*	*Description*
type	Integer	Is 4 if successful or 0 in case of a generic error
code	Integer	Assumes a value from the Response Codes table
services	Array of Service	Array of available services (in the specified format), can be and empty array in the event of no available services

LIST Response Example

```
{"type":4, "code":0,"services":[{"id":14,"name":"Operator Service 5Mbps", "type":"EF",
"bandwidth":5000}, {"id":16,"name":"Operator Service 2Mbps", "type":"EF",
"bandwidth":2000}]}
```

17.1.5 Run Request

Run Request generates a list of running service sessions. It returns a dataset containing the session list.

The List Request session and message format is shown in the tables below.

Table 55 VPS Engine: Run request, session format

Field	**Type**	**Description**
id	String	Alphanumeric string that identifies the service session
service	Integer	Base Service ID
bandwidth	Integer	Service Speed in kbps
source	String	IPv4 of the source point
destination	String	IPv4 of the destination point
start	String	UTC Data and Time of the session start

Table 56 VPS Engine: Run request, message format

Field	***Type***	***Description***
key	String	API Key
type	5	Request type, must be 5

RUN Request Example

```
{"key":"ECE335024E3E466CA98BF5014D5C7D86", "type":5}
```

The format and an example of the data set response to a Run Request message are shown below.

Table 57 VPS Engine: Run response

Field	**Type**	**Description**
type	Integer	Is 5 if successful or 0 in case of a generic error
code	Integer	Assumes a value from the Response Codes table
sessions	Array of Session	Array of running session (in the specified format), can be and empty array in the event of no running session

RUN Response Example

```
{"type":5, "code":0
"sessions":[{"id":"710.1.3.110.1.1.10.448130380713861241347963976614","service":6,
"bandwidth":150, "source":"10.1.3.1","destination":"10.1.1.1","start":"2012-09-18
```

10:26:18"},{"id":"710.1.3.110.1.1.10.74715744255735061347973738228","service":7,
"bandwidth":100, "source":"10.1.3.1","destination":"10.1.1.1","start":"2012-09-18 13:09:00"}]}

17.2 Response Codes

Some codes might not appear for some request responses.

Table 58 VPS Engine: Response codes

Code	**Meaning**
0	Request executed successfully
1	Invalid API Key
2	Unknown Request
3	Invalid arguments
4	Invalid service
5	Invalid session
6	Insufficient bandwidth available (on modify it is considered that the service remains running unaltered)
7	No Path between source and destination points with requested service type
8	Unable to execute operation (internal failure)
9	Nothing to Modify (trying to modify to running service)
10	Source-Destination pair already being used in another service

17.2.1 Generic Error Responses

Some errors are generic in nature and do not conform to a specific type. These errors will be returned as response of type 0, with the error code.

Generic Error Example

```
{"type":0, "code":1}
```

18 Appendix 4: Portal database

The Portal uses an SQL database as backend. This appendix includes the current definition of the database tables, using SQL dialect supported by PostgreSQL.

```
CREATE TABLE users (
  id                integer PRIMARY KEY,
  username          text UNIQUE NOT NULL,
  email             text UNIQUE NOT NULL,
  profiles          integer -- Flags: app developer, device maker, MNO, researcher
);

CREATE TABLE apps (
  id                integer PRIMARY KEY,
  code              text UNIQUE NOT NULL,
  user_id           integer NOT NULL FOREIGN KEY REFERENCES users (id),
  name              text NOT NULL,
  os                integer -- Enum: Android, iOS, Other
);

CREATE TABLE app_versions (
  id                integer PRIMARY KEY,
  app_id            integer NOT NULL FOREIGN KEY REFERENCES apps (id),
  version           text NOT NULL,
  version_code      text,
  file              bytea,
  CONSTRAINT unique_version UNIQUE (app_id, version)
);

CREATE TABLE app_user_flows (
  id                integer PRIMARY KEY,
  app_id            integer NOT NULL FOREIGN KEY REFERENCES apps (id),
  kpi_id            integer NOT NULL FOREIGN KEY REFERENCES kpis (id),
  name              text NOT NULL,
  script            bytea NOT NULL
);

CREATE TABLE kpis (
  id                integer PRIMARY KEY,
  name              text NOT NULL,
  description       text,
  kpi_type          integer -- Enum: app, device...
);

CREATE TABLE kpi_marks (
  id                integer PRIMARY KEY,
  kpi_id            integer NOT NULL FOREIGN KEY REFERENCES kpis (id),
  name              text NOT NULL,
  description       text,
  order             integer,
  allowed_types     integer, -- Flags: user action, UI element, UI element
property, internal
  internal_help     text
);

CREATE TABLE kpi_mark_supports (
  id                integer PRIMARY KEY,
  app_user_flow_id  integer NOT NULL FOREIGN KEY REFERENCES app_user_flows (id),
  kpi_mark_id       integer NOT NULL FOREIGN KEY REFERENCES kpi_marks (id),
  support_type      integer NOT NULL, -- Enum: user action, UI element, UI element
property, internal
  user_action       integer,
  ui_element        text,
  ui_element_property   text
);

CREATE TABLE scenarios (
  id                integer PRIMARY KEY,
  name              string UNIQUE NOT NULL
);

CREATE TABLE test_configurations (
  id                integer PRIMARY KEY,
  scenario_id       integer NOT NULL FOREIGN KEY REFERENCES scenarios (id),
  name              string UNIQUE NOT NULL
)

CREATE TABLE devices (
  id                integer PRIMARY KEY,
  string            name UNIQUE NOT NULL,
  owner_id          integer FOREIGN KEY REFERENCES users (id),
  testbed           boolean NOT NULL,
  os                integer -- Enum: Android, iOS, Other
);

CREATE TABLE campaigns (
  id                integer PRIMARY KEY,
  owner_id          integer NOT NULL FOREIGN KEY REFERENCES users (id),
  name              text NOT NULL,
  certification     boolean NOT NULL,
  campaign_type     integer, -- Enum: app, device, mno, researcher
  repeat            integer NOT NULL,
  state             integer -- Enum: created, scheduled, running, finished
);

CREATE TABLE campaign_app_versions (
  id                integer PRIMARY KEY,
  campaign_id       integer NOT NULL FOREIGN KEY REFERENCES campaigns(id),
  app_version_id    integer NOT NULL FOREIGN KEY REFERENCES app_versions(id),
  CONSTRAINT unique_campaign_app_version UNIQUE (campaign_id, app_version_id)
)

CREATE TABLE campaign_kpis (
  id                integer PRIMARY KEY,
  campaign_id       integer NOT NULL FOREIGN KEY REFERENCES campaigns (id),
  kpi_id            integer NOT NULL FOREIGN KEY REFERENCES kpis (id),
  app_user_flow_id  integer FOREIGN KEY REFERENCES app_user_flows (id),
  CONSTRAINT unique_support UNIQUE (campaign_id, kpi_id)
);

CREATE TABLE campaign_scenarios (
  id                integer PRIMARY KEY,
  campaign_id       integer NOT NULL FOREIGN KEY REFERENCES campaigns (id),
  scenario_id       integer NOT NULL FOREIGN KEY REFERENCES scenarios (id),
  CONSTRAINT unique_scenario UNIQUE (campaign_id, scenario_id)
```

```
);

CREATE TABLE campagin_devices (
  id                 integer PRIMARY KEY,
  campaign_id        integer NOT NULL FOREIGN KEY REFERENCES campaigns (id),
  device_id          integer NOT NULL FOREIGN KEY REFERENCES devices (id),
  CONSTRAINT unique_device UNIQUE (campaign_id, device_id)
);

CREATE TABLE experiments (
  id                 integer PRIMARY KEY,
  campaign_id        integer NOT NULL FOREIGN KEY REFERENCES campaigns (id),
  order              integer,
  app_version_id     integer FOREIGN KEY REFERENCES app_versions (id),
  kpi_id             integer FOREIGN KEY REFERENCES kpis (id),
  app_user_flow_id   integer FOREIGN KEY REFERENCES app_user_flows (id),
  scenario_id        integer FOREIGN KEY REFERENCES scenarios (id),
  device_id          integer FOREIGN KEY REFERENCES devices (id),
  oml_database       text
);
```