Hindawi
Mobile Information Systems
Volume 2019, Article ID 2563917, 8 pages
https://doi.org/10.1155/2019/2563917

Research Article

An End-to-End Automation Framework for Mobile Network Testbeds

Almudena Díaz Zayas, Bruno García García, and Pedro Merino

University of Malaga, Málaga, Spain

Correspondence should be addressed to Almudena Díaz Zayas; adz@uma.es

Received 18 January 2019; Accepted 6 March 2019; Published 1 April 2019

Guest Editor: Tara A. Yahiya

This paper describes the end-to-end automation framework developed as part of the TRIANGLE project. The TRIANGLE project is devoted to the benchmarking of apps and devices in mobile networks. For that purpose, it is needed to ensure the repeatability in the behaviour of all the components of the mobile network during the execution of the same test. This is why one of the main objectives of the project was to develop an end-to-end automation framework to provide repeatable testing. This paper describes in detail the design and the implementation of the framework.

1. Introduction

Conformance to standards [1] at the radiofrequency level is a legal requirement to allow the deployment of new mobile devices in the (regulated) mobile networks. Since the origins of 2G networks, Europe has led the creation and exploitation of conformance testing procedures and tools for mobile devices. The major vendors of smartphones apply testing methods to certify that their devices comply with the radio and signalling standards and are authorized to operate in the regulated spectrum.

However, the certification of radio and protocol signalling related features does not guarantee a good behaviour of the mobile devices when running applications, and this could be a huge problem considering the critical role apps will play in the 5G era. Underperformance of apps or unexpected failures due to the interaction of different apps in the same device could be detected with proper testing and qualification methods. The main objective of the TRIANGLE project is to develop the TRIANGLE testbed [2] for the testing of applications and the performance of devices when running apps.

In this context, the project focused on the development of a framework for testing applications in a set of repeatable scenarios on reference handsets. This means a fully controlled environment (from the network up to the handset) for which the elements behave in a reproducible manner which allows identifying differences generated by or due to the application under test. The controlled environment will allow the emulation of different radiofrequency scenarios and backend network impairments. The outcome of the test is a benchmark score comparing the application to a set of reference values and defined KPIs (key performance indicators) [3].

Current solutions for testing apps with different networking conditions are based just on software tools to emulate the provision of specific network qualities in terms of latencies or error rate (some examples are Facebook's Augmented Traffic Control [4] or Dymmynet [5]). Such software emulation means an advance with respect to pure simulation [6, 7] of the whole system in a computer because they support the connection of the real devices as part of the end-to-end path. However, the network itself is just an illusion. Emulating 5G network effects like seamless handover or changes in the quality of a bearer dynamically can hardly exhibit the same behaviour than real network equipment. Furthermore, the behaviour of the mobile device running the application should require a lot of (not always feasible) modifications to be connected to such software emulated network.

The main objective of the TRIANGLE testbed is to provide realistic 5G behaviours to unmodified commercial mobile devices. For this purpose, the testbed combines actual industrial 4G/5G testers, commercial EPCs, commercial mobile user equipment, etc. to offer emergent networking scenarios in a very flexible and programmable environment that ensures repeatability of experiments. With this approach, commercial devices can be directly connected to the platform to run the apps in the selected networking scenarios.

The main building blocks of the TRIANGLE testbed architecture are the testing framework and the testbed infrastructure, composed by hardware and software components. The testing framework covers all the software, coordination/sequencing that automates the control and management of the testbed infrastructure. It is in charge of handling and converting the test requests into actionable steps within the software and hardware portion of the testbed. This paper focuses on the description of the TRIANGLE testing framework, whose main target is the full automation of the underlying software and hardware infrastructure of the testbed.

The most relevant contributions of the paper are the description of the methodology used to implement the TRIANGLE testing framework and the introduction of the Test Automation Framework (TAP) from Keysight, a modern Microsoft.NET-based application that can be used for the development of advanced automation software for testing systems. The methodology is inspired by the testing methodology used in the telecommunication domain that is based on the specification of a set of test cases which contains the description and configuration of the testing environment and the actions to be executed during each test. This methodology ensures that the testing is repeatable, regardless of the equipment and the entity performing the certification. That means that the same test cases are adopted and implemented by a different testbed, and the results provided by both of them should be comparable. The TAP tool enables to translate the test cases into executable TAP test plans. The TAP test plans allow automating the configuration, control, and execution of the tests.

Section 2 describes the testbed infrastructure. Section 3 introduces the TAP software and additional components used to build the so-called TRIANGLE testing framework. The interaction between the testing framework and the testbed infrastructure is explained in Section 4; this section details the control interfaces implemented by the testing framework to manage each one of the components of the testbeds. Finally, Section 5 remarks the main outcomes of the TRIANGLE testbed.

2. Testbed Infrastructure

In order to provide a better understanding on how the testing framework performs, this section provides an overview of the different components which integrate the testbed.

The TRIANGLE 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 applications or devices and to provide test reports. On top of the infrastructure layer, operates the testing framework which fully automates the configuration, control, and execution of the test cases specified in the project [8] for the testing of applications and devices.

Figure 1 shows an overview of the physical interconnections between the testbed components.

The commercial mobile devices are connected to the UXM Wireless Test Set from Keysight [9]. This equipment is used traditionally in the conformance testing of mobile devices. The UXM plays the role of RAN (radio access network) in the TRIANGLE testbed. Some of its key features of the UXM are flexible intercell interference coordination (eICIC) schemes, WLAN (wireless local area network) offloading, and IMS (IP multimedia subsystem)/end-to-end VoLTE (voice over LTE) communications between multiple devices.

The radio signal is not radiated (over-the-air); instead, it is conducted through calibrated RF cables to the UE antenna connector. For testing purposes, most UEs typically contain small antenna connectors which are hidden from the user. The UXM supports the interconnection of two UEs. To connect more devices to the same UXM, the testbed uses RF switches, controlled by a switch driver. The switches are placed in the RF connection between the UEs and the UXM, and the switch driver will select which RF connections (RF paths) to be routed to the UXM.

The UXM emulates all the network signalling and physical signals, including MIMO (multiple input multiple output) configurations. All the protocol layers in the emulated network operate realistically as defined in the 3GPP (3rd Generation Partnership Project) test specifications and can be configured. Moreover, for testing purposes, the UXM instrument provides additional useful capabilities, such as a downlink channel emulator to emulate the effect of actual radiated propagation, detailed logging, and friendly control.

The battery pins of the UE (user equipment) are connected to the power analyzer N6705B. This allows both the control of the input voltage into the phone and to measure the instantaneous current drawn by the device. The N6705B power analyzer supports up to four devices connected at the same time.

In addition, the mobile 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 (Android Debug Bridge) tool. To support several mobile devices, the testbed use a DUT USB hub.

The following elements are connected via Ethernet in a local network: UXM, N6705B, switch driver, management server, core network, and transport. The testbed also includes a virtualized infrastructure based on OpenStack to support the local deployment of services.

The core network is a commercial EPC (evolved packet core) from Polaris Networks, which includes the main elements of a standard core network: MME (mobility management entity), SGW (serving gateway), PGW (packet data network gateway), HSS (home subscriber server), and PCRF (policy and charging rules function), which have been

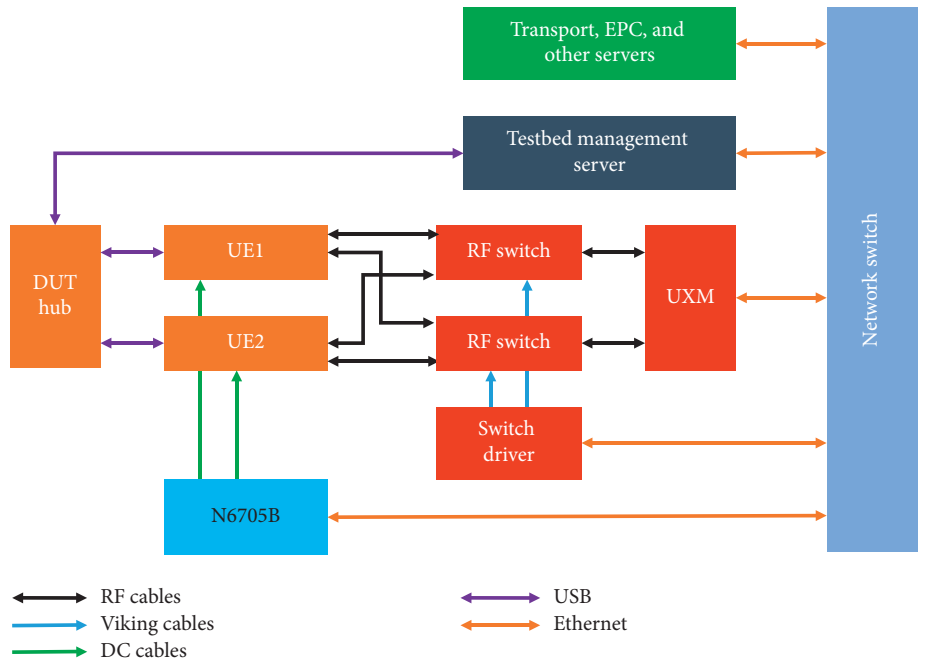

Figure 1: TRIANGLE testbed infrastructure.

integrated. In addition, this EPC includes the ePDG (evolved packet data gateway) and ANDSF (access network discovery and selection function) components for dual connectivity scenarios.

To emulate the transport network between the eNodeB and the EPC, TRIANGLE uses an SDN (software defined networking) 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 offers the possibility of integrating artificial impairments in the interfaces of the core network and the application servers.

Two software probes are deployed at the UE, the DEKRA TACS4 agents [10], and the TestelDroid tool [11] from the University of Malaga. TRIANGLE also provides an instrumentation library for application developers, in order to provide additional measurements that cannot be extracted by other means. There are additional software probes running at the core network and at the backhaul. Hardware probes include a power analyzer connected to the UE to measure power consumption and the probes available in the radio access emulator. All the probes are also under the control of the testing framework.

Finally, in the management servers, there is deployed a software tool called Quamotion WebDriver [12]. This tool is based on the Selenium technology used for web testing [13]. Quamotion WebDriver has adapted this technology for the testing of Android applications. This tool is also under the control of the testing framework to run the tests.

3. Triangle Testing Framework

The TRIANGLE testing framework has been developed in the scope of the TRIANGLE project. A key enabler for the automation implemented in the testing framework is TAP. 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: Leveraging C# and Microsoft Visual Studio; TAP is a platform upon which it is possible to build test solutions.

Particularly, TAP has been used for the implementation of the test cases defined in the context of the TRIANGLE project [8] and the implementation of the interfaces to control the components of the testbed infrastructure. This section focuses on the implementation of the test cases, while Section 4 focuses on the description of the plugins developed to automate the control of the testbed infrastructure.

TAP allows the definition of test plans. TAP test plans consist of a sequence of test steps, which specify the set of actions that are performed on different TAP instruments (the logical entities that control the physical equipment on the platform) or control the execution flow of the test plan (for instance, repeating certain steps or executing different actions depending on the results of a previous test step). TAP defines a set of basic test steps that are suited for most user requirements, and it is possible to develop custom steps in C# for complex or very specific needs.

TAP instruments, and the functionally equivalent DUTs (devices under test), define the logic for interacting with other entities in the facility. TAP includes a generic instrument for controlling SCPI (standard commands for programmable instruments) compatible equipment, while custom TAP instruments can be developed using C#. Each TAP instrument encapsulates the configuration and management logic of the equipment, as well as defining the actions that can be performed on it.

TAP plays a key role in the implementation of the testing framework as it allows controlling and automating all the components present in the testbed. In the context of TRIANGLE, a test plan is composed by the sequence of steps specified by the test cases. In each test step, there is a set of configurable parameters. The value of the configurable parameters can be made available externally in the test plan; hence, it is possible to modify these values before the execution of a test plan, making possible to orchestrate the execution of TAP using an external entity. In this way, it is possible to use TAP for controlling the fine-grained interaction with the different equipment of the platform, while keeping a separate, upper layer for general test case orchestration.

Python has been used to implement the logic on top of TAP that decides the specific values to be configured in the test plan. The decision is based on the information stored in the YAML configuration files. The YAML files contain the values of the parameters specified in the test cases.

Python is a modern programming language with a large community of developers that continues to improve with the inclusion of new built-in modules and features on each subsequent release. Additionally, there is a large amount of ready to use, open source, frameworks, and libraries suited for common needs. This can facilitate the development of the components of the platform, for example, by using Python's standard libraries on the definition of the REST API (representational state transfer application programming interface) that supports the communication between different components.

Since Python is an interpreted language, it is also possible to modify the code rapidly, with no need for a separate compilation stage on a development machine, which can drastically reduce the complexity of testing and the time needed for the resolution of simple issues.

Figure 2 shows a more detailed view of the main functional blocks that make up the TRIANGLE testing framework. The architecture can be divided into several subsystems, whose role will be introduced briefly in this section.

End users access the TRIANGLE testbed through the Portal [14]. In this Portal, they can upload the applications under test. In addition, they will have to declare the features or capabilities of their applications. These features will define what can be tested and which test case specifications will be applicable during the testing of the application. In addition to the application features, end users will have to provide additional information; for example, they have to choose the device on which the application will be executed during the test and the scenario (network and propagation conditions, for example, pedestrian or vehicular scenarios).

In order to improve the utilization of the TRIANGLE testbed, it is possible to queue several campaign executions at a time. These campaigns will be executed successively by the testbed, avoiding idle periods of time while waiting for a user to start the execution of another campaign. This has been achieved by means of a scheduler layer between the TRIANGLE portal and the Orcomposutor.

The ORchestrator-COMPOSer-execUTOR (Orcomposutor), implemented in Python, is a server with a REST API that runs on the same Windows machine as TAP.

Once all the required information has been entered in the Portal, the TRIANGLE testbed end user can proceed with the test. The first step would be to take the collected information and turn it into executable TAP test plans. This is the task of the test plan Orcomposutor. According to the features introduced in the Portal for the application under test, the Orcompusutor will generate the applicable test plans.

To create the required TAP test plans, the Orcomposutor uses predefined TAP test plan templates. The TAP test plan plays the same role that the TTCN3 implementations of the test cases defined by the 3GPP for the certification of mobile protocols implementation [15]. When possible, the Orcomposutor will take advantage of two TAP features: the ability to expose parameters of a test step to external callers and a test step that allows the execution of another test plan. Figure 3 shows an example of test plan. The test plan is composed by a test plan reference which contains the network configuration of the test. The rest of test steps are used to configure and control the components of the testbed. In the right side of Figure 3, the configurable parameters exposed by the test steps are shown.

For instance, many test plans will start by setting up the network scenario and configuring the required parameters in the testbed equipment. This setup is the same, regardless of the body of the test plan. Thus, the Orcomposutor reuses existing TAP test plans that configure particular network scenarios.

For an application test, the body of the test plan typically includes replaying the user actions contained in an application user flow provided by the application developer. The Orcomposutor gets the application user flow from the Portal and sets the corresponding external parameter of the WebDriver replay test step. WebDriver is the tool used to interact automatically with the application under test (click on buttons, scroll).

The Orcomposutor is also aware of which KPIs are going to be measured with each of the generated TAP test plans. If necessary, the test plan should provide explicit support for performing the measurements required for the KPIs. For instance, if a test plan will contribute to a KPI on power consumption, the power analyzer must be configured and used in the test plan.

Each of the TAP test plans created by the Orcomposutor can then be executed in the Testbed using TAP. The TAP test plan contains all the information required to execute a test automatically.

During the execution of the TAP test plan, the measurement tools will gather measurements. The measurement tools that are fully integrated with TAP will publish them as usual. In this case, the results will be handled by a TAP result listener that sends them to a central OML server [16]. This OML server uses a PostgreSQL database server to store the measurements. Some tools may include OML support, and thus send their measurements directly to the OML server.

The main functions of the Orcomposutor can be summarized as follows:

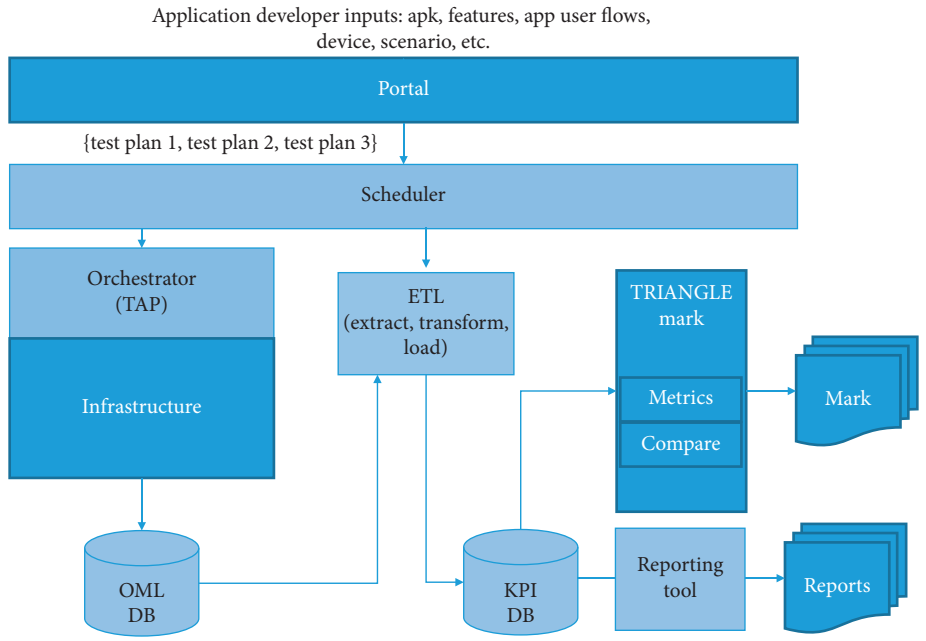

Figure 2: TRIANGLE testing framework architecture (light blue boxes).

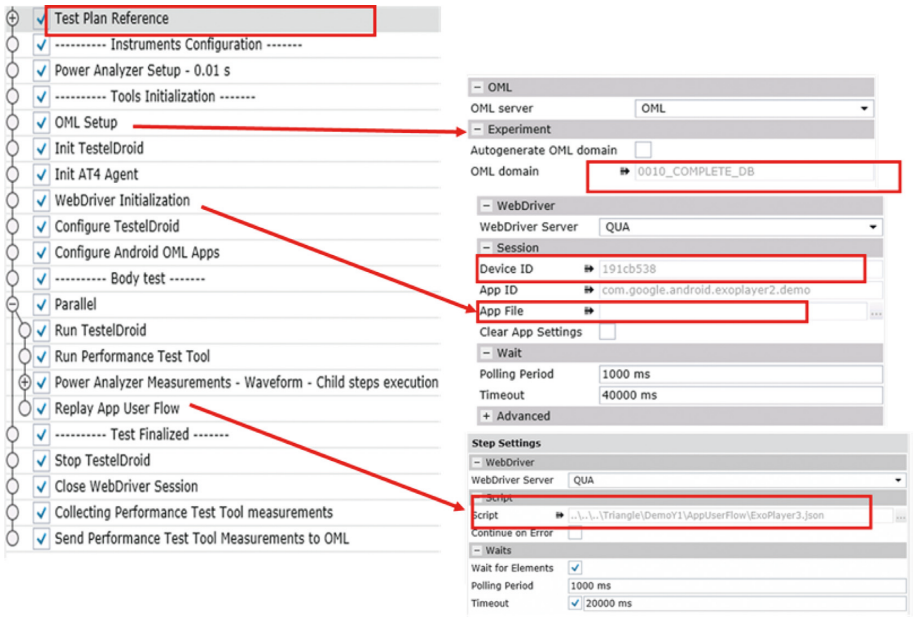

Figure 3: TAP test plan template for executing TRIANGLE test cases.

(i) Accept test campaign execution requests from the Portal.

(ii) Compose the TAP test plans required to run a test campaign and its test cases

(iii) Execute the TAP test plans

(iv) Upload the results of the execution to the Portal

To carry out these functions, Orcomposutor needs to communicate with the REST API of the Portal and with the OML database.

The request to execute a test campaign only includes the identifier of that campaign. Orcomposutor uses that id to request more information about the test campaign to the backend using its REST API. This information is used to determine which test case or test cases must be executed with TAP. A test campaign might include more than one test case; in that case, the Orcomposutor will prepare the execution of more than one TAP test plan.

TAP test plans contain several external parameters and test plan references that must be filled in before execution

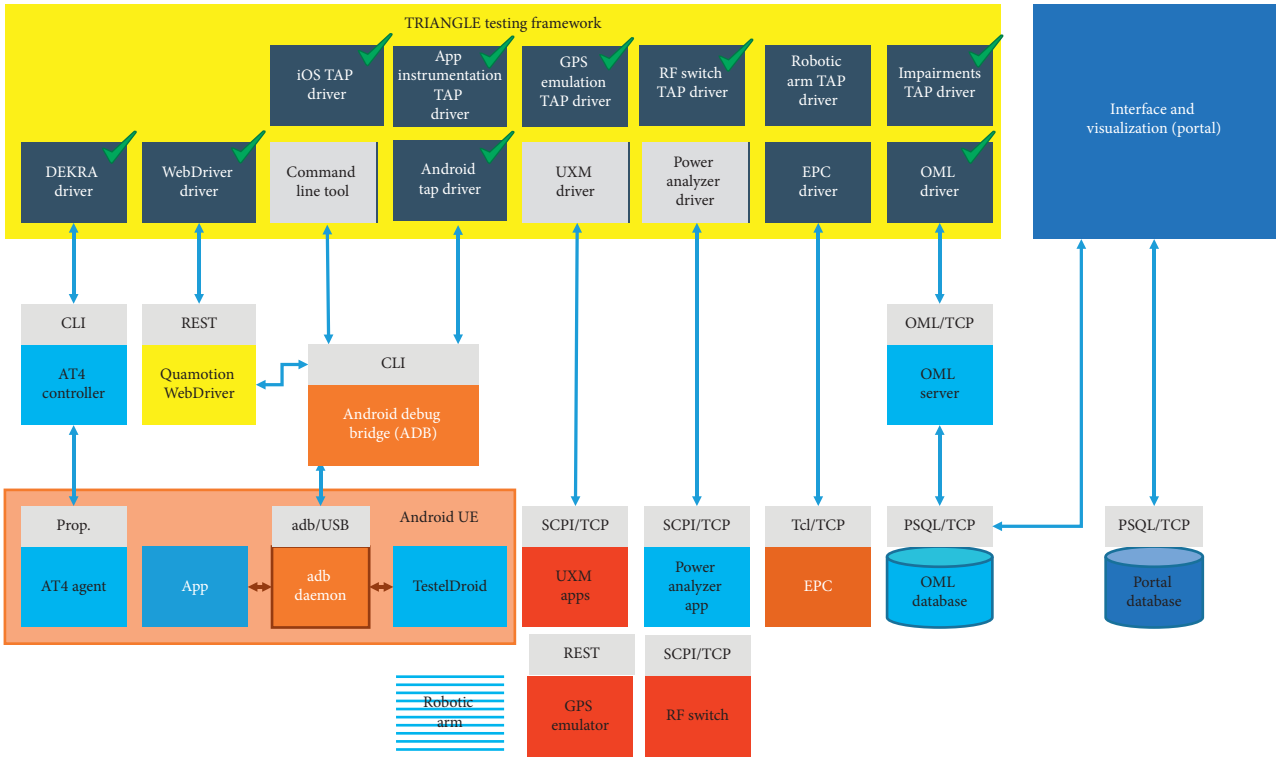

Figure 4: Testbed infrastructure automation based on TAP plugins.

can start. Orcomposutor must retrieve the appropriate information from the backend REST API in order to fill in these blanks (Figure 3), such as the id of the device used in the test case or the network scenario. We call the selection of this parameters and referenced TAP test plans the composition of the test plan.

Once the TAP test plan has been composed, it can be executed with the TAP CLI (command line interface). The Orcomposutor will store the TAP logs, as well as internal logs for diagnostic purposes.

4. Testbed Automation

In this section, we introduce the TAP plugins implemented to control each one of the components of the testbed. The plugins contains the test steps used in the TAP test plan shown in Figure 3. In Section 3 was introduced TAP and its usage as sequencer of actions to configure and control the components. Figure 4 shows the software interfaces (TAP plugins) developed to communicate the testing framework and the infrastructure components.

In TAP, an instrument is a logical entity that encapsulates the interaction with a physical instrument. At the very least, a TAP instrument must define all the necessary logic for connecting and disconnecting the TAP host machine with the real instrument, and it is a best practice to define methods for performing every possible (or required) actions that the instrument can execute.

For example, a TAP instrument for controlling a real power supply via SCPI must include two methods (open and close) that create and release the connection with the instrument and can have two extra methods for setting the voltage and current.

Instruments are extensively used in TAP steps, which expose the instrument functionality to the end user. Continuing with the previous example, we could define two steps for this instrument: one that turns off the power output, setting voltage and current to 0, and another that sets the voltage and the current using the values selected by the user in the test step configuration. TAP will automatically use the open and close methods from the instrument when required.

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 more components. 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 are delivered through a TCP connection.

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 the Android device and perform user actions, such as tapping or swiping. These commands are delivered to the Android UE using ADB. TestelDroid is also managed through ADB.

DEKRA TACS4 tool is managed through the proprietary control interface provided by the tool.

The EPC can be controlled through a fixed set of TCL (tool command language) scripts that use a TCL scripting API provided by the EPC components emulators. The TAP driver will execute these scripts, which will then send the appropriate commands to the EPC through the TCL API.

The measurements from all the probes are collected and sent to a central OML (ORBIT measurement framework and library) 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 (comma-separated values) 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.

5. Conclusions

One of the main contributions of the TRIANGLE project is the design and development of a testing framework that automates the end-to-end configuration and control of a mobile communication testbed. The testing framework is based on TAP (test automation platform), a powerful editor of test sequences that also includes an SDK for the development of plugins for components offering a control interface.

The testing framework has proven to be flexible and sustainable by integrating a large number of components different in nature such as RAN (radio access network) equipment, a core network, instruments for measuring power consumption, mobile devices, and software tools acting as probes.

Moreover, the TRIANGLE testing framework is going to be adopted in the 5GENESIS project [17] in the experiment life cycle manager component of the coordination layer of the 5GENESIS facility. 5GENESIS is a project devoted to the realization of a 5G platform that will allow the validation of the major 5G key performance indicators (KPIs). 5GENESIS project is integrated by five platforms: Athens, Berlin, Limassol, Malaga, and Surrey.

Data Availability

This paper is about one of the main outputs of the Triangle project, the end-to-end automation of the TRIANGLE testbed. More detailed information about the the project can be found at https://www.triangle-project.eu/project-old/deliverables/. In particular, automation is described in WP3 deliverables. UMA is the WP3 leader and the partner in charge of the automation task.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The TRIANGLE project was funded by the European Union's Horizon 2020 Research and Innovation Program (grant agreement no. 688712). The 5GENESIS project was funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 815178.

References

[1] W. Lin, H. Zeng, H. Gao, H. Miao, and X. Wang, "Test sequence reduction of wireless protocol conformance testing to internet of things," *Security and Communication Networks*, vol. 2018, Article ID 3723691, 13 pages, 2018.

[2] A. F. Cattoni, G. C. Madueño, M. Dieudonne et al., "An end-to-end testing ecosystem for 5G the TRIANGLE testing house testbed," *Journal of Green Engineering*, vol. 6, no. 3, pp. 285–316, 2016.

[3] A. D. Zayas, L. Panizo, J. Baños, C. Cárdenas, and M. Dieudonne, "QoE evaluation: the TRIANGLE testbed approach," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 6202854, 12 pages, 2018.

[4] B. Blywis, M. Guenes, F. Juraschek, and J. H. Schiller, "Trends, advances, and challenges in testbed-based wireless mesh network research," *Mobile Networks and Applications*, vol. 15, no. 3, pp. 315–329, 2010.

[5] A. Zubow and R. Sombrutzki, "A low-cost MIMO mesh testbed based on 802.11n," in *Proceedings of the 2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3171–3176, Paris, France, April 2012.

[6] R. Marco Alaez, J. M. Alcaraz Calero, Q. Wang et al., "Open-source based testbed for multioperator 4G/5G infrastructure sharing in virtual environments," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 1984314, 11 pages, 2017.

[7] C.-K. Jao, C.-Y. Wang, T.-Y. Yeh et al., "WiSE: a system-level simulator for 5G mobile networks," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 4–7, 2018.

[8] EU H2020, "TRIANGLE project, deliverable D. 2.2 final report on the formalization of the certification process, requirements and use cases," https://www.triangle-project.eu/project-old/deliverables/, 2017.

[9] E7515A UXM, "Wireless test set getting started guide," 2017, https://www.keysight.com/main/gated.jspx?lb=1&gatedId=2459161&cc=US&lc=eng&parentContId=2371785&parentContType=ct&parentNid=-32909.0&fileType=VIEWABLE.

[10] TACS4 Performance Testing Platform, https://performance.tacs4.com/.

[11] A. Álvarez, A. Díaz, P. Merino, and F. J. Rivas, "Field measurements of mobile services with Android smartphones," in *Proceedings of the 2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 105–109, Las Vegas, NV, USA, January 2012.

[12] Quamotion WebDriver, http://docs.quamotion.mobi/webdriver/.

[13] A. Bruns, A. Kornstadt, and D. Wichmann, "Web application tests with Selenium," *IEEE Software*, vol. 26, no. 5, pp. 88–91, 2009.

[14] A. Díaz-Zayas, A. Salmerón Moreno, G. García Pascual, and P. Merino Gómez, "TRIANGLE portal: an user-friendly web interface for remote experimentation," in *Smart Industry & Smart Education*, M. Auer and R. Langmann, Eds., Springer, Cham, Switzerland, 2019.

[15] ETSI Standard ES 201 873-1 V3.4.1, The Testing and Test Control Notation Version 3; Part 1: TTCN-3 Core Language, 2008, Sophia Antipolis, France, European Telecommunications Standards Institute (ETSI).

[16] M. Singh, M. Ott, I. Seskar, and P. Kamat, "ORBIT measurements framework and library (OML), motivations, design, implementation, and features," in *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2005)*, Trento, Italy, February 2005.

[17] H. Koumaras, D. Tsolkas, G. Gardikis et al., "5GENESIS: the genesis of a flexible 5G facility," in *Proceedings of the 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 17–19, Barcelona, Spain, September 2018.