F-Interop Platform and Tools: Validating IoT Implementations Faster

Maria-Rita Palattella, Federico Sismondi, Tengfei Chang, Loïc Baron, Malisa Vucinic, Pablo Modernell, Xavier Vilajosana, Thomas Watteyne

F-Interop Platform and Tools: Validating IoT Implementations Faster*

Maria Rita Palattella[1], Federico Sismondi[2], Tengfei Chang[3], Loic Baron[4], Mališa Vučinić[5], Pablo Modernell[6], Xavier Vilajosana[6], and Thomas Watteyne[3]

[1] Luxembourg Institute of Science and Technology, Luxembourg
mariarita.palattella@list.lu
[2] IRISA, Campus de Beaulieu, Rennes, France
federico.sismondi@irisa.fr
[3] Inria, Paris, France
{tengfei.chang,thomas.watteyne}@inria.fr
[4] Sorbonne Université, LIP6, Paris, France
loic.baron@lip6.fr
[5] University of Montenegro, Podgorica, Montenegro
malisav@ac.me
[6] Universitat Oberta de Catalunya, Barcelona, Spain.
{pmodernell0,xvilajosana}@uoc.edu

Abstract. F-Interop allows implementors of IoT protocols to test compliance and interoperability of their implementation by connecting to a remote server in the cloud which contains and runs testing tools for different IoT standards. This paper provides an overview of the F-Interop Platform and the tools. Nowadays testing tools for CoAP, 6TiSCH, OSCORE and LoRaWAN standards are already available.
The F-Interop project, which has started as a H2020 project funded by the European Commission, is now supported by a large community of standard development organizations, small/large companies, academic institutions, and protocol designers and implementors. They all see the benefits of adopting the new Interoperability approach proposed by F-Interop, which allows to validate IoT implementation faster, reducing time to market, and standardization.

Keywords: IoT · interoperability · online tools · standardization.

1 Introduction

The principal barrier to massive IoT technology adoption is the lack of interoperability and the resulting segmented nature of the IoT market. The classical approach for checking conformance to a given standard, and interoperability

* Supported by the H2020 F-Interop project (https://www.f-interop.eu/), and in particular the Open Call projects SORT, SPOTS, and F-LoRa.

with other products from different vendors, consists in organizing interoperability events, Face-to-Face (F2F) meetings where vendors meet and run interoperability tests. The F2F approach has a set of drawbacks, in term of cost to afford, and timeline to launch a standard-based and interoperable product on the market. Participants must cover engineering and travel costs, and this is often not possible for SMEs. During the event, there is usually not enough time for vendors to run all the tests, or to fix an implementation when a test fails. To test their product again, vendors must wait for the next event (which produce a delay in the time to market) and also invest new resources, to attend the new Interop meeting.

To overcome such drawbacks, the H2020 F-Interop project has fostered the development of online testing tools for emerging IoT standards and technologies, accessible remotely, on a cloud platform [1][2]. The shift in the Interoperability approach, from the classical F2F events to the online and remote ones, has several benefits: (i) ease the Interop procedure, using online tools, (ii) drastically reduce cost - no need to travel, only remote attendance, and (iii) faster the time to market, i.e., the production of standard-based and interoperable products.

In this paper, we describe the F-Interop Platform, and the integrated testing tools for different IoT technologies (CoAP, 6TiSCH, LoRaWAN, etc.). We share the lessons learned from the first F-Interop Interop events entirely run adopting the new approach.

2 F-Interop Platform

The F-Interop platform developed as a *platform as a service* offers remote online testing tools for IoT protocols. It has been designed with a modular and scalable architecture which allows easy integration of new testing tools. As illustrated in Fig. 1 it includes a set of core components, described hereafter, which communicate among them using an Event Bus.

Session Orchestrator: It is the core component of the architecture. It is in charge of provisioning (and tearing down) a virtual host per test session within the Event Bus in order to isolate the tests running suimultaneously. It is also responsible for supervising the processes associated with the testing tools, e.g. spawning and/or killing the testing tools containers on-demand, logging testing tool's log output, etc.

Resource Repository: It is the component responsible for the storage and retrieval of the resources used in the tests. These resources are selected by the user when launching the test session. Examples of resources include the user Implementation Under Test (IUT), virtual IUTs available on the Platform to run conformance tests, testbeds and related IoT devices, etc.

Result Repository: It is the component in charge of storing and retrieving the results generated during F-Interop test sessions.

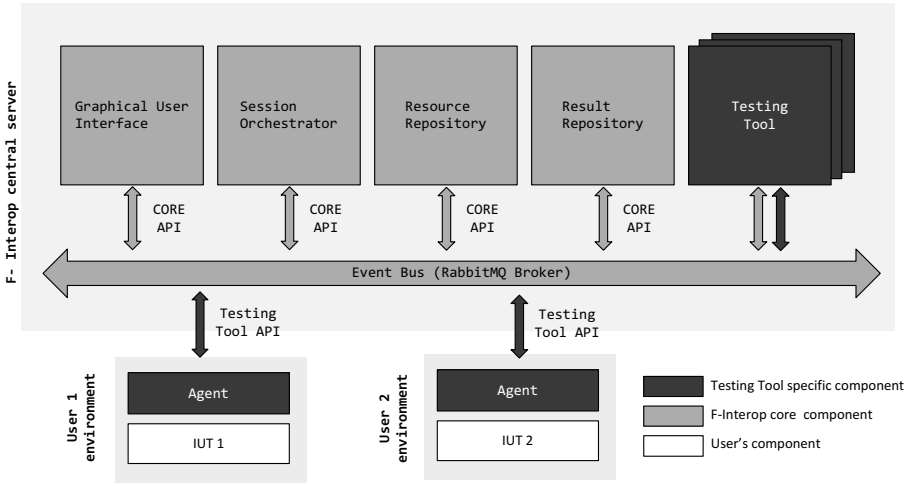

Fig. 1. F-Interop Platform and CORE components.

Testing Tool: It is a protocol-specific component. It is responsible for the interaction between the user components and the F-Interop platform during the test execution. It is also in charge of analyzing the network traffic exchanged between IUTs, and to issue a verdict based on the tests results. The Testing Tools communicates over the Event Bus, and the messages exchanged must comply with the defined API. F-Interop supports several testing tools for different IoT protocols (CoAP, 6TiSCH, OSCORE, LoRAWAN, etc.) as detailed in the following sections of the paper.

Graphical User Interface (GUI): It integrates the different components of the F-Interop platform together, offering to the end users an easy access through the web. Anyone with a web browser can create an account on the F-Interop Platform. Once logged in, the user can launch a new test session in three simple steps: (i) select the testing tool of interest, (ii) configure the tool according to the IUT features, and (iii) start the test session. Users can save the configuration setting in a JavaScript Object Notation (JSON) format for later re-use, during new test sessions. Once running, the testing tool sends messages to the GUI explaining to the user the steps to follow in order to complete the tests.

Agent: It is the proximity component of the Testing Tool which runs in the user environment close to the IUT. It can be a program which runs in the user's operating system, or a specific firmware which runs in a specific hardware platform.

Event Bus: It is the common medium used for exchanging all type of messages (control messages, raw data packets, logs, etc.), among the different components of the F-Interop platform. Each message sent over the Event Bus is composed of a routing key, properties and a body. The routing key indicates how to route a message to the relevant input queues of the listening components. The properties include a message identifier, a time-stamp defining the date and time when the message has been produced and a content-type specifying the use of the JSON format. It can also present a reply-to field and a correlation-id identifier in case of a request/reply exchange of messages. RabbitMQ is the underlying message-passing mechanism. It acts as a secure message broker between all the components through encrypted channels.

CORE API: It standardizes a set of messages which are sent over the Event Bus for providing the basic platform services such as "`display message to user`" (GUI), "`save test results information`" (Results Repository), and more. The complete feature set of the CORE API can be accessed online[7].

3 F-Interop testing tools

This section describes the testing tools currently integrated in the F-Interop Platform, addressing some of the most relevant IoT protocols: CoAP, 6TiSCH, OSCORE and LoRaWAN.

3.1 CoAP Testing Tool

The Standard: CoAP, the Constrained Application Protocol, standardized by the IETF CoRE working group, is the de-facto web transfer protocol for constrained devices. It is based on a request-response interaction model, and its basic architecture includes a CoAP client and a CoAP server.

CoAP runs on the connection-less unreliable transport protocol UDP. Therefore, it implements at application layer some lightweight reliable mechanisms for detecting duplication, re-ordering and acknowledging delivery. Orthogonal to these protocol features, CoAP defines also the GET, POST, PUT and DELETE methods in RFC7252. Then RFC7641 extends the feature set with the OBSERVE mechanism: CoAP clients can "observe" states of resources from CoAP servers (in other words it can retrieve the value of a resource, and get its updates over a given period of time).

The Test Description: All the aforementioned protocol features and mechanisms can be tested with the F-Interop testing tool. The CoAP reference interoperability Test Description, TD[8] has been integrated into F-Interop[9].

[7] `doc.f-interop.eu#events-core-api`
[8] `https://github.com/cabo/td-coap4`
[9] `http://doc.f-interop.eu/testsuites/coap`

The Testing Tool: The CoAP Testing Tool offers to end users a controlled and feature-rich environment that eases the execution of the online and remote standard-based interoperability test procedure.

Fig. 2 shows the setup for running a CoAP remote interoperability test between two IUTs. Two protocol-specific components (the Agent and the Testing Tool) have been implemented for supporting CoAP tests.

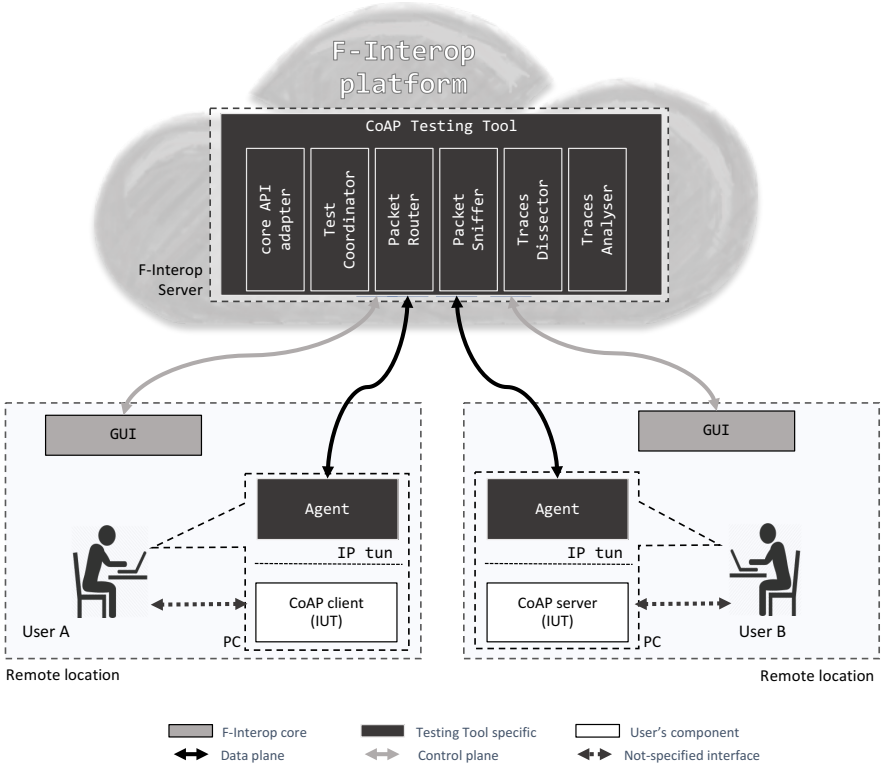

Fig. 2. CoAP remote user-to-user interoperability test setup.

The main components of the CoAP Testing Tool are:

1. `Packet Router and Agents:` establishes a VPN-like setup between IUTs. The controlled environment helps users bypass UDP-blocking firewalls and other middle boxes installed in their facilities. The setup creates IPv6 network interfaces bound to the VPN, therefore users can test their implementations over IPv6 regardless the IP version supported by their internet provider. The Packet Router is the middle-box between IUT1's interface and IUT2's interface, it can act as a lossy gateway for test scenarios which require simulating a lossy context.

2. `Test Coordinator:` coordinates the entire interoperability test. It iterates over the test steps described in the test description. It dispatches commands to users through the GUI, based on the TD (e.g. ‘‘`user1: CoAP Client is requested to send a GET request`’’). The user is guided to perform the test remotely.
3. `Packet Sniffer:` sniffs the traffic exchanged between IUTs and generates PCAP files records. The component enables the export feature for network traces so users can analyze the exchanged frames using tooling outside of F-Interop e.g. wireshark or some analysis scripts code.
4. `Traces Dissector:` dissects the exchanged messages between IUTs and provides a human readable representation of the packets. It provides a wireshark-like view to helps users find problems encountered during the interoperability test execution.
5. `Traces Analyzer:` analyzes the traffic exchanged between IUTs during a test case. The tool automatically issues PASS, FAIL, INCONCLUSIVE verdicts after each test case. The analysis is based on the CHECK steps of the test cases description.

F-Interop Integration: The `core API adapter` module handles the integration with F-Interop's core services such as GUI and Results Repository.

The integration enables two interoperability test use cases: (i) remote user-to-user interoperability session, as described in Fig. 2, and (ii) single-user sessions to test interoperability against reference implementations.

3.2 6TiSCH Testing Tool

The Standard: The 6TiSCH protocol stack, designed and standardized at IETF, glues together the IEEE802.15.4 PHY and MAC protocols with an IPv6-based upper stack. The latter includes the RPL routing protocol, the 6LoWPAN adaptation protocol, and the CoAP application protocol. Thanks to the Time Synchronized Channel Hopping (TSCH) scheme adopted at the MAC layer, 6TiSCH can offer low-latency, high reliability, scalability, and low power consumption, thereby meeting the requirements of Industrial Internet of Thing (iIoT) applications. 6TiSCH has defined the minimal configuration to run a 6TiSCH network (RFC8180), and a 6P sublayer that manages the allocation of MAC resources.

The Test Description: The first 6TiSCH Interoperability TD was prepared for the first 6TiSCH Plugtest organised by ETSI in 2015, at IETF93 [3]. Since then, the TD has been updated several times, for each new ETSI Plugtest and Interop event organised in the framework of the F-Interop project. The latest version of the TD is available online [10] and includes four classes of tests: `SYN` tests (testing the synchronization status of device), `MINIMAL` tests (testing the

[10] http:/openwsn-builder.paris.inria.fr/job/td-6tisch%20builder/lastSuccessfulBuild/artifact/6tisch_plugtest_td_june_2018.pdf

minimal configuration of 6TiSCH), `6LoRH` tests (testing IP packets in 6LoRH format, according to RFC8138), and `6P` tests (testing format and behaviors of the 6P sublayer).

The Testing Tool: The 6TiSCH testing tool is composed of three main protocol-specific components:

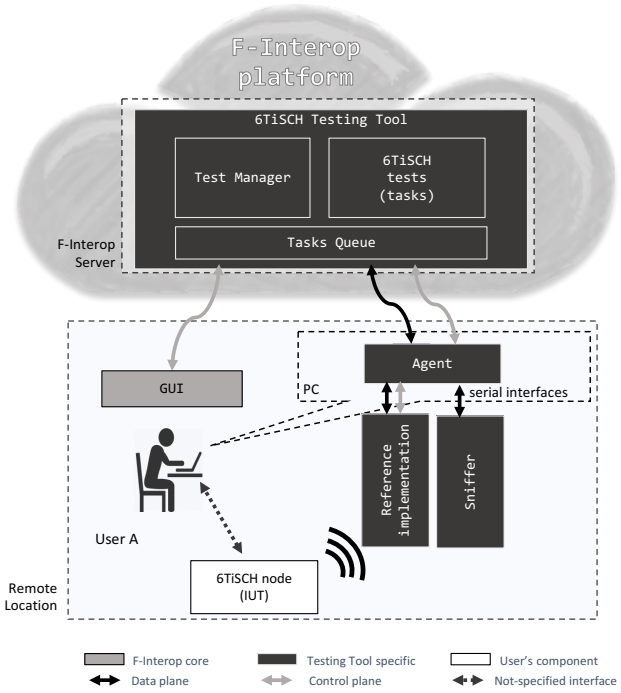

Fig. 3. 6TiSCH remote conformance test (IUT vs reference implementation) set up.

1. `Test Manager:` It is the brain of the 6TiSCH testing tool. To start the 6TiSCH manager, the user needs to select the 6TiSCH suite on the web GUI and start a session. Once it is running, the manger sends a message to the GUI asking which specific 6TiSCH test the user would like to execute. An AMQP consumer is up and running and listens to the choice made by the user. Once this information is received, the manager creates a test context which includes the sequence of steps required by that specific test, according to TD.
2. `6TiSCH Tests (tasks):` It is the component dispatching STIMULI actions to users (start/shut-down IUT), controls to the packet sniffer (start/stop),

and handling CHECK steps analysis after the test execution, using the Wireshark dissector.

3. `Task Queue:` It is in charge of executing the tasks generated by the manager and/or the 6TiSCH Tests (tasks) component. Once a task is received, the task queue assigns it to one of its workers that is currently idle and marks it as busy. Hence multiple tasks can be assigned at the same time using several workers. Once the task is completed, then the worker can get back to the idle status. The Task Queue is implemented in Celery [11].

F-Interop Integration: Fig. 3 shows the general setup for running a 6TiSCH conformance test using the F-Interop Platform. While running a test, the Manager generates three types of tasks for the Task Queue: `step start`, `step end`, `testcase verdict`. For all the tests, besides the specific feature for which interoperability is checked, the user has to perform some generic steps: turn on all the devices (the IUTs, and the packet sniffer); wait until the devices enter into a stable state; issue a stimulus to let the device perform a given action. As final step, to check the result of the test, the user uploads the Wireshark packet capture (pcap) file, which is analyzed by the Wireshark dissector. If the packet is found in the pcap file, and all the check fields are correctly dissected, then the manager gives a PASSED verdict. If the packet cannot be found or dissected, the test fails. The specific reasons why the test failed are shown inside the red bottom verdict.

In future developments of the 6TiSCH tools, an `Agent` component will be integrated to interact with the IUT and automatize some steps of the test session. This includes the initial execution of the stimulus (e.g. send a 6P ADD packet), and the final upload of the pcap file. Currently, these operations are executed manually by the end user.

3.3 OSCORE Testing Tool

The Standard: The Object Security for Constrained RESTful Environments (OSCORE) is a mechanism that provides application-layer security of CoAP. OSCORE secures CoAP by authenticating and encrypting the CoAP message and its payload, and by cryptographically bounding responses to requests. OSCORE excludes from cryptographic processing parts of the CoAP message that are supposed to be read or modified by a proxy. As a consequence, OSCORE enables CoAP to be used through untrusted intermediaries. This contrasts with the use of (D)TLS to secure CoAP, where any such intermediary needs to terminate the secure connection, and has full access to the exchanged data. OSCORE also enables the use of secure CoAP for group communication.

OSCORE relies on a pre-established security context between a client and a server. This security context encompasses cryptographic key(s) and initialization vectors, as well as application-level identifiers. The OSCORE security context

[11] `http://www.celeryproject.org/`

can also be agreed upon dynamically, through an independent key exchange protocol.

OSCORE has been specifically designed for constrained IoT devices. It is therefore very efficient in terms of message overhead, and is an appealing solution for providing a secure end-to-end channel. As such, OSCORE has been adopted as a key security component in the IETF 6TiSCH protocol stack. Apart from securing CoAP application exchanges, in a 6TiSCH network, OSCORE also secures the "join" process of a new node.

The standardization work around OSCORE has first started in late 2014, and the specification is currently in the last stage before being published as an Internet Standard. There are multiple implementations of the draft standard available, including 2 in Python, 1 in C# and 2 in embedded C. As part of the F-Interop project, OSCORE has also been implemented within Wireshark, to facilitate packet analysis.

The Test Description: The core OSCORE team has released several versions of the OSCORE Test Description[12], used during F2F plugtest events held in the past, typically co-located with IETF meetings. In the framework of the SPOTS project, funded by the F-Interop Open Call, the TD has been redesigned to facilitate automated testing.

The OSCORE test description encompasses a total of 16 tests. These tests are divided into 3 categories: setup tests verifying the interoperability of the underlying CoAP implementation; correct usage tests, covering basic OSCORE functionality; incorrect usage tests, covering error handling behavior.

F-Interop Integration: The OSCORE testing tool has already been integrated in the F-Interop Platform. A video demonstration of an OSCORE test is available online [13]. It complements the 6TiSCH testing tool. From the point of view of an F-Interop user, the steps described in Section 3 are followed as in case of any 6TiSCH test: queries prompting for different configuration parameters, test stimuli, and the upload of a pcap file of a test execution. In case a user selects an OSCORE test, the GUI simply prompts the user for additional OSCORE-specific parameters, such as the application identifiers and cryptographic keys. This implementation method enables the testing of OSCORE on top of the 6TiSCH protocol stack, as well as any other generic OSCORE implementation whose packet capture the user can obtain. In the latter case, the 6TiSCH specific parameters are simply bypassed during the configuration prompt.

The OSCORE test suite relies on the Wireshark implementation to decrypt and dissect the packet(s). The Wireshark output is parsed and compared to the values expected by the test case semantics. For example if the decryption and authentication code verification of an OSCORE message was successful. Ongoing implementation work focuses on the integration of the OSCORE suite with the

[12] `https://github.com/EricssonResearch/OSCOAP`
[13] `https://youtu.be/05-MTLrFSrg`

CoAP testing tool, in order to make it possible to launch an OSCORE test from the CoAP testing tool, leveraging the VPN-like tunneling provided by the Agent component.

3.4 LoRaWAN Testing Tool

The Standard: LoRaWAN is one of the most adopted Low-Power Wide Area Networking (LPWAN) technologies, with a simple network architecture that offers a multi-km communication range provided by the robustness of a LoRa physical layer. In the LoRaWAN network architecture, end devices communicate with application servers using gateways and a central network server, in a star-of-stars topology. The end devices communicate with the gateways using a single hop LoRa radio link. The gateways forward the received messages to the network server using, for example, an IP network.

Although the communication is bi-directional, uplink communication from the end devices to the network server is privileged. Three end device classes are defined: A, B, and C. The selection of the device class is based on a compromise between energy consumption and latency. Class A devices, mandatory to implement per the current LoRaWAN specification, defines two downlink reception windows after every successful uplink transmission in which the device will listen for downlink messages.

The LoRa PHY layer uses pseudo-random channel hopping and each frame is transmitted using a Spreading Factor (SF) in a trade-off between communication range and the time-on-air of the frames. The network server is in charge of the configuration of the different communication parameters in order to comply with the duty-cycle regulations of the used frequency band.

LoRaWAN implements a two-layer security mechanism. One key is used for the end-to-end encryption of the messages exchanged between the end device and the application server (this key is not known by the network). A second key is used for integrity checking and the encryption of the MAC commands exchanged by the end device and the network server. The MAC commands are used by the network server to configure the communication parameters of the nodes (e.g., selected data rate, delay of the downlink reception windows, data rate used in the reception windows).

The Testing Tool: Considering the LoRaWAN network architecture, the F-LoRa testing tool is designed to play the role of the Network Server and Application Server while interacting with an end device. As shown in Fig. 4, the end device is running the LoRaWAN Implementation Under Test (IUT) and the user must provide a compliant LoRA gateway running a packet forwarder.

The Test Description: A test description has been defined for testing the main specification of the LoRaWAN protocol. The tests are classified into different groups, based on the type of features they aim to verify: (i) ACT, device

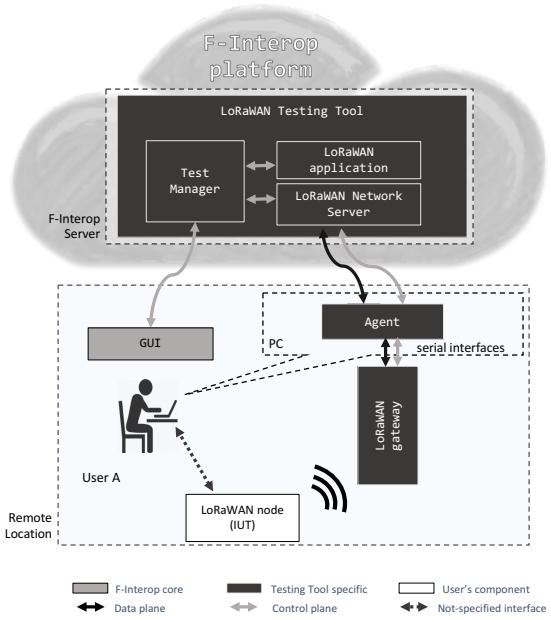

Fig. 4. LoRaWAN remote interoperability test set up.

activation, (ii) FUN, basic functionalities and timing, (iii) SEC, security, encryption and integrity check, (iv) MAC, MAC commands. The different activation mechanism, Activation By Personalization (ABP) and Over the Air Activation (OTAA) can be tested. In addition to the basic joining message exchange of the OTAA, new data rate, reception windows delay, and frequencies are configured using the Join Accept message to test the implementation of this feature. Regarding security, end-to-end encryption with the configured keys can be tested and the Message Integrity Code calculation can be verified. Between the different messages exchanged with the IUT, new configuration parameters are set using MAC commands to check that the IUT behaves as expected.

F-Interop Integration: The user interacts with the F-LoRa testing tool using the F-interop GUI. After the selection of the F-LoRa LoRaWAN testing tool, instructions are provided on how to download, run, and configure the Agent module on the user's side. The ABP credentials of the IUT (security keys, short address, device EUI) can be configured and then the user selects a group of tests to be performed.

The Agent serves as an interface between the gateway's packet forwarder and the testing tool. The user must configure the gateway to send all the packets to the Agent component. Once the test session has started, all interactions are handled by the testing tool. The testing application running on the device with the IUT responds as defined by the testing protocol. A detailed description of

the test verdict, indicating the errors found in any failing step is shown to the user in the GUI after all tests are completed.

4 From F2F to online remote Interop events

The SORT project, funded by the 1st F-Interop Open Call, has largely promoted the adoption of the F-Interop Platform and testing tools, organizing two F-Interop 6TiSCH Interop events. During the first event, organized in July 2017, and co-located with IETF99-Prague, the IETF community was informed about the H2020 F-Interop project and the possibility to run Interop tests online, using the F-Interop Platform. The use of the Platform was demonstrated with the set of 6TiSCH tools available at that time. Members of standard development organizations, small/large companies, academic institutions, and protocol designers and implementors, showed interest in adopting the F-Interop Platform, and run the tests online. This was confirmed by their attendance of the 2nd F-Interop 6TiSCH Interop event, held in Paris in June 2018. 15 people from 10 different organizations attended each of the two F-Interop event. Participants could test their implementations and validate the results of the tests, uploading a pcap file on the F-Interop platform. Participants run not only 6TiSCH tests, but also CoAP and OSCORE tests. The adoption of the F-Interop Platform by a wide Interop community (6TiSCH, CoAP, OSCORE) shows the feasibility of the initial F-Interop project industrialization plan.

6LoRITT-P&F and CoAP-P&F are two newly founded projects by the 2nd F-Interop Open Call which will kick-off during the second half of 2018. In the same spirit of the SORT project, they will organize remote Interop events for CoAP and 6LoWPAN. Participants located in different places around the globe will be connecting to F-Interop plaform to run Interop tests. To cope with the difference in working hours between different geographical locations, the Interop events will take place in a series of time slots, time-shifted between each occurrence in an efficient way for covering all participants local times requirements.

As already proved by the first F-Interop Interop events, IoT academic and industrial players, including SMEs, will benefit from the F-Interop Platform, as they will be able to check compliance and interoperability of their products in a shorter timeframe, and with highly reduced cost.

References

1. Leone, R., Sismondi, F., Watteyne, T., Viho, C.: Technical Overview of F-Interop. In: 2nd EAI International Conference on Interoperability in IoT (InterIoT), Paris, France (2016)
2. Eunsook, E. K., Ziegler, S.: Towards an open framework of online interoperability and performance tests for the Internet of Things. In: Global Internet of Things Summit (GIoTS), Geneva, Switzerland (2017)
3. Palattella, M.R., Vilajosana, X., Chang, T., Ortega, M.A., Watteyne, T.: Lessons Learned from the 6TiSCH Plugtests. In: EAI Int. Conf. on Interoperability in IoT (InterIoT), Rome, Italy, (2015).