F-INTEROP

HORIZON 2020
Information and Communication Technologies
Integrating experiments and facilities in FIRE+

Deliverable D2.2
Complementary Conformance Test Enablers - 1st Iteration

Grant Agreement number:	**687884**
Project acronym:	**F-Interop**
Project title:	**FIRE+ online interoperability and performance test tools to support emerging technologies from research to standardization and market launch** **The standards and innovations accelerating tool**
Type of action:	**Research and Innovation Action (RIA)**
Project website address:	**www.finterop.eu/**
Due date of deliverable:	**M12**
Dissemination level:	**PU**

This deliverable has been written in the context of the Horizon 2020 European research project F-Interop, which is supported by the European Commission and the Swiss State Secretariat for Education, Research and Innovation. The opinions expressed and arguments employed do not engage the supporting parties.

Co-funded by the European Union

Co-funded by the Swiss Confederation

Document properties

Responsible partner	**INRIA**
Author(s)/editor(s)	Remy Léone, Thomas Watteyne, Federico Sismondi, César Viho
Version	**1.0**
Keywords	Interoperability Testing, Conformance Testing, Remote Testing, Online, Platform, Testing components, Test enablers

Abstract

This report corresponds to the deliverable: **D2.2 - Complementary Conformance Test Enablers 1st iteration**.

The deliverable **D2.2** is the first release of the online conformance test core enablers. It is the first version of the enablers for online remote conformance testing which complements the key enablers described in deliverable D2.1. It includes new methods and/or adaptations of existing conformance testing tools and additional components needed for online remote conformance testing.

Table of Contents

List of Figures

List of Acronyms

6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
CoAP	Constrained Application Protocol
EC	European Commission
ETSI	European Telecommunications Standards Institute
EU	European Union
GPS	Global Positioning System
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communication Technologies
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Standards Organization
IT	Information Technology
MAC	Media Access Control
OS	Operating System
R&D	Research & Development
SME	Small Medium Enterprise
TAT	Test Analysis Tool
TED	Test Extended Description
TL	Task Leader
URL	Uniform Resource Locator
WP	Work Package
W3C	World Wide Web Consortium

1 Introduction

The deliverable D2.2 is the first version of the enablers for online remote conformance testing which complements the key enablers described in D2.1. To avoid any duplication, you can refer to the introduction of the D2.1 which gives you a good overview of the objectives of the WP2.

2 Complementary conformance test enablers

2.1 F-Interop testing architecture and components

In this section the F-Interop testing architecture is presented in Fig. 1. The components of this architecture are responsible for managing the testing infrastructure necessary, including provisioning the underlying network, capturing trace, starting/stopping the different tests, and reporting the verdicts. Through standard security mechanisms, the architecture ensures the authentication of the different users, and the confidentiality of test results.

2.1.1 The "Event Bus" Software Design Pattern

The F-Interop architecture is composed of different components exchanging messages through an "Event Bus". All communication is done through this mechanism, including control messages, raw data packets and logs. We use RabbitMQ as the underlying message-passing mechanism. It acts as a secure message broker between all the components through encrypted channels.

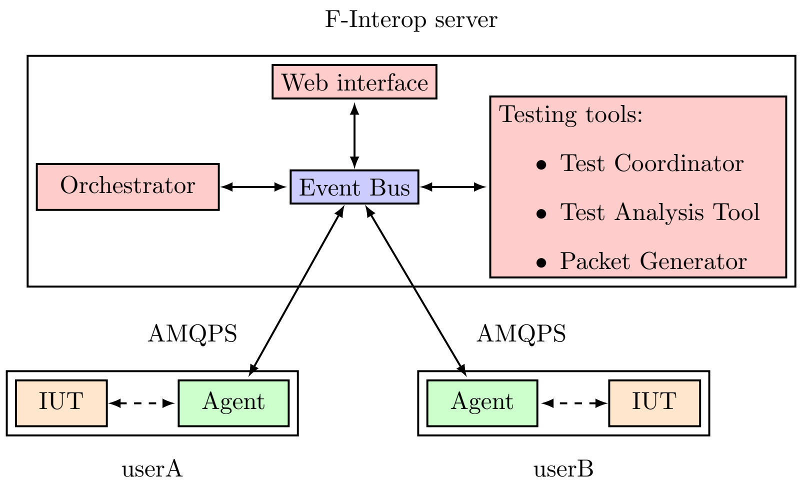

Figure 1 – F-Interop remote interoperability and conformance testing architecture

Each message contains a routing key and a topic which indicates how to route this message to the relevant input queues of the components. Messages are of two types: ***control plane*** and ***data plane***. The ***control plane*** messages relate to the management of an ongoing test session: e.g. start a sniffer, signal the start/end of a test case, etc. The ***data plane*** messages contain the raw data exchanged between the IUTs.

The AMQP (Advanced Message Queuing Protocol) standard is agreed as being an interesting solution for message passing. RabbitMQ [1] implements AMQP and enables differentiated ACL security/authorization profiles. AMQP bindings are available in several languages. They can be used for remote interaction with the IUTs. Orchestration and distinct channels can be used for distinct testing tools.

A F-Interop-User conducts remote interoperability and conformance tests in independent sessions. We use the virtual host mechanism of RabbitMQ[1] to ensure isolation between concurrent sessions.

This architecture is modular and scalable by design. Components can be added/deleted from the event bus without requiring further coordination. Different components can be run on different (virtual) machines to ensure scalability. Different components can be written in different programming languages. The only thing that they need to do is being able to send AMQP messages to the F-interop platform.

Because of that, new complementary tools can be added both on the F-interop side and on the user side. The upcoming sections present such additional components.

2.2 F-interop side additional components

The **testing tool** handles information coming from the IUT throughout a F-Interop session. They can be started once the different users are connected and the necessary components are provisioned by the orchestrator. The role of the testing tool is to generate verdicts that corresponds to test cases and the packets exchanges that happened during the session. The F-Interop platform does not impose a particular design for the testing tool. They operate as a black box which emits coordination messages towards agents, and generates test verdicts. It is typically composed of a test coordinator, a test analysis tool and a packet generator (mainly for conformance testing).

2.2.1 Conformance testing using a protocol dissector

Protocol dissector are a testing tool that dissect network packets to produce a description of the different packet fields. Therefore, a dissector can be used in conformance testing to assert that a given implementation under test implement correctly a given protocol.

Implementing a dissector can be long and tedious if the protocol is complex, therefore it is a good strategy to use popular dissectors that already dissect many protocols layer instead of starting everything from the beginning.

Also, contributing to an already existing dissector project is an efficient way to get feedback on the code quality of such dissector as well and use efficient tools that are already deployed and used massively.

Therefore, for 6TiSCH, we choose to use the official Wireshark code repository and add our dissector to it in order to get as much feedback and support from the community as possible.

Once a dissection is done, we get a report in the form of a JSON document containing all the field of the packet in an easy to handle form. Once a packet is in its form it's fairly easy to test the data it contains and perform conformance testing on it.

This process can be repeated for any new protocol and can be applied to new protocols coming from the Open Call for instance.

2.2.2 Container-based testing tools

New functionalities can be added easily to the F-interop platform due to the message oriented architecture. Such functionalities might have dependencies that are not compatible with other testing tools. Managing it individually for each tool would be cumbersome.

Containers [7] can bundle all those dependencies and simply expose a network interface that would connect those lightweight virtual machines to the F-interop server.

Therefore, using containers is a good approach to have a lightweight virtualization that is both easy and efficient to use because those containers would contain all the required dependencies.

During the Open Call, contributors would simply have to send a container with their own tools, their own dependencies. The F-interop platform would have simply to start such container and connect them to the F-interop platform for events to be exchanged and additional features enabled.

2.3 User side additional components

The modular architecture of the F-interop platform also allows us to add new features on the user side. More specifically around the agent which is the software made to work with the F-interop and connect the users to it. Here is a quick overview of the new enablers for the agent component.

2.3.1 Connecting the agents with new software: Agent hooks

Agent needs to be able to interact with a large set of heterogeneous devices. Adding functionality on the agent side for every IUT would bloat the agent with too many particular use-cases.

Instead of doing so, agents are bundled with hooks that can be used by the F-interop platform to trigger them remotely and making them trigger particular behaviour of IUT using a network protocol (HTTP, ping) or a more low-level way (serial port).

For instance, let's suppose that an IUT would be triggered to perform a certain action if it receives an HTTP request. The user would specify it on the F-interop side, then the test coordinator would at the right moment order the agent to send such HTTP request.

By doing so, in case of automated testing, a user would simply have to give indication about how to trigger its IUT by using those hooks. Those hooks are platform agnostics and are configured by the user depending on his/her particular test setup.

2.3.2 Additional agent feature

Some additional feature allows the agent to improve the accuracy or the ease to perform some tasks. The next sections present such improvement:

2.3.2.1 Precise time measurement

Some protocols such as 6TiSCH requires very precise timestamping to ensure that synchronization is performed reliably. GPS time synchronization can be used to get precisely time measurement and therefore perform the synchronization tests with enough precisions.

Agent can interface with such process easily using hooks and upload all the observation to the F-interop platform once the observation is done.

2.3.2.2 Easy deployment

Agent will be deployed on various platform and they will use different libraries to perform the different tasks they are required to do. As so, the agent might be bundled as a single binary to easy up the deployment on new testbeds and user’s computers.

Once the agent downloaded and started, it would download additional bundles from the F-interop platform to perform certain tasks: for instance flashing a new mote or update itself.

3 Status of work

As it has been previously mentioned, the work related to T2.1 and T2.2 has been carried out in parallel due to the mutualized F-Interop testing architecture. In the following sub-sections, we present some indicators on the status of work of the first iteration of Complementary conformance test enablers (corresponding to the deliverable D2.2). As said previously, these components are based on 6TiSCH's interoperability and conformance testing suite experiences.

3.1 Status of work for T2.2 – Deliverable D2.2

The Reference-based interop testing has been considered as an easy way to identify components that are needed for remote conformance online testing. Actually, to move to conformance testing only the component implementing the golden device (the testing tool) has to be replaced by the conformance testing tool together with a Packet generator.

As mentioned previously, the use-case of 6TiSCH is used to develop a first version of Complementary conformance test enablers (corresponding to the deliverable D2.2). The 6TiSCH test suite implements several conformance tests (in particular for the 6P protocol [3, 4]).

6TiSCH test cases demonstrate how F-interop can be used to perform tests in low-level network layer such as the link layer. Tests aims to verify for instance that two devices can synchronize with each other.

A typical test in 6TiSCH uses a reference implementation called a golden device that connect to the computer host through a visualizer [5] which create a virtual network interface to connect the 6TiSCH network to a local area network. Once a 6TiSCH network is set-up, the visualizer can send commands to each of the 6TiSCH node inside it through a serial port in order to trigger behaviour such as the emission of a specific packet required by a given test.

Our development effort currently focuses on connecting the 6TiSCH reference implementation (Golden device) with the F-interop agent. This connection is agnostic to a specific protocol and uses message sent from the agent towards the golden device to perform the remote control of a device through a serial port. This connection is done using ZeroMQ[6] sockets between the visualizer and the agent. We currently can send and receive any messages that the visualizer manages.

To summarize, the work accomplished for the first iteration of the T2.2 uses most of the components used in T2.1 (Agent, Orchestrator, testing tool, Test coordinator, Web interface, Test analysis tool) that allow to perform remote online interoperability testing for 6TiSCH. On the 6TiSCH focused development, we've been the following progress:

- First implementation of the message exchange between the event bus, the test coordinator, the agents and the 6TiSCH reference implementation. The emission of a 6TiSCH packet by the reference implementation can be triggered by the F-interop platform remotely.
- Enhancement done to a 6TiSCH packet dissector used during the analysis of the network traces. This dissector was successfully integrated to official Wireshark code repository making its dissemination way more efficient and streamlined.

4 Conclusion and next steps

This document presented the work related to the task T2.2 that has been carried out in parallel with the task T2.1 due to the mutualized F-Interop testing architecture. We present some indicators on the status of work of the first iteration of Complementary conformance test enablers (corresponding to the deliverable D2.2). We first presented the components of the F-Interop testing architecture that are responsible for managing the testing infrastructure necessary, including provisioning the underlying network, capturing trace, starting/stopping the different tests, and reporting the verdicts. The outcome of the work achieved in this period is the development of the first version of these components based on CoAP interoperability tests suite and 6TiSCH's interoperability and conformance testing suite experiences. It includes the development of a first version of the following components: The Agent, Orchestrator, testing tool, Test coordinator, Web interface, Packet generator, Test analysis tool). They allowed performing remote online reference-based testing for 6TiSCH.

By this way, these components correspond to the first iteration of Complementary conformance test enablers (corresponding to the deliverable D2.2).

The work to be done in next steps includes:

- Completing the development of the components of the online remote testing architecture
- Implementation more tests for 6TiSCH
- Integration of the testing architecture to the testbeds (FIT IoT-Lab, iMinds, etc.) allowing executing remotely online interoperability with components on those testbeds
- Extending testing tools for including Web of Things (WoT) interoperability testing tools.

5 References

[1] https://www.rabbitmq.com/

[2] http://supervisord.org/

[3] https://tools.ietf.org/html/draft-wang-6tisch-6top-protocol-00

[4] https://www.ietf.org/mail-archive/web/6tisch/current/pdfgDMQcdCkRz.pdf

[5] https://openwsn.atlassian.net/wiki/display/OW/OpenVisualizer

[6] http://zeromq.org http://zeromq.org

[7] https://www.docker.com/what-docker