Technical Overview of F-Interop

Rémy Leone[1], Federico Sismondi[2], Thomas Watteyne[1], César Viho[2]

[1] Inria, EVA team, France.
first.last@inria.fr
[2] Irisa, France.
first.last@irisa.fr

Abstract. Interoperability and conformance testing are needed to ensure that systems behave as specified by the standards they implement. Today, interoperability testing is done through face-to-face "interop events". Requiring physical presence of all parties impacts the scalability of the testing, and slows down the development of standards-based products.
F-Interop is a platform which enables *remote* interoperability and conformance testing of networking standards. This paper gives a technical overview of the project and its software architecture. The architecture follows the event bus design pattern: generic messages are routed between the different software components, some of these running at different locations.

Key words: Interoperability Testing, Conformance Testing, Remote Testing, Online, Platform.

1 Introduction

F-Interop is a platform which provides remote interoperability and conformance testing of network standards. F-Interop allows to reduce the time to market of devices by providing a platform to test interoperability remotely and autonomously to find problems sooner. It also helps communities working on standards finding at an early stage potential interoperability problems in draft standards.

This paper gives a technical overview of the F-Interop platform represented, and serves as a technical companion paper to [5]. Its software architecture which will be described in detail throughout this paper.

The remainder of this paper is organized as follows. Section 2 presents current best practice and the associated limitations. Section 3 introduces the F-Interop platform with a focus on the technical architecture. Section 4 presents how a test is executed in the platform. Section 5 discusses how this architecture is suitable for many types of test cases.

2 Interoperability and Current Best Practice

Conformance testing determines whether a system complies to the requirements. Conformance testing is key for having interoperable implementations, but it is not enough on its own. For this reason, conformance testing is always complemented with interoperability testing. Interoperability testing focuses on end-to-end functionality between two systems/implementations implementing the same standard(s).

Both conformance and interoperability testing are based on use cases which are abstract illustrations of the typical behavior of a system. The behavior is defined in standards, a document (usually a standard or technical specification) from a recognized Standards Developing Organization (SDO). A Test Description (TD) is derived from the standard. It is a set of test cases which covers the different behavior a standard defines. The goal of conformance and interoperability tests is to run test cases, and for each generate a pass/fail verdict.

Today, interoperability events are face-to-face meetings in which vendors bring their Implementation Under Test (IUT). The TD of the event is prepared before the events and distributed to the participants. The TD contains a list of Test Cases (TC), each of them describing a particular configuration and a sequence of actions the participants need to follow. ETSI[1] has for example organized interoperability events for various low-power wireless protocols such as CoAP [4, 1], 6Lo(WPAN) [2]. and 6TiSCH [3].

Interoperability Test Description			
Identifier:	TD_COAP_CORE_01		
Objective:	Perform GET transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.1, 4.4.3, 5.8.1		
Pre-test conditions:	• Server offers the resource /**test** that handle GET with an arbitrary payload		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is requested to send a GET request with: • Type = 0(CON) • Code = 1(GET)
	2	check (CON)	Sent request contains Type value indicating 0 and Code value indicating 1
	3	check (CON)	Server sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content type option
	4	verify (IOP)	Client displays the received information

Fig. 1. Example CoAP test case, as specified in [1].

Table 1 gives an simple example test case for the CoAP protocol, as specified in [1]. For this test case, one CoAP client IUT issues a CoAP `GET` request

[1] The European Telecommunications Standards Institute, `http://www.etsi.org/`.

(the "stimuli") to a CoAP server IUT. The CoAP Server is pre-configured to offer resource `/test`. A sniffer mechanisms is required to capture the different messages exchanged. Once the CoAP transaction is over, participants then manually check the format/contents of these messages, and verify that they comply with the standards (steps 2 and 3 in Table 1). The test case generates a "pass" verdict if all the "check" steps pass and the users verify that their IUTs behaved correctly.

Hundreds of face-to-face interoperability events have taken place, resulting in numerous standards compliant and interoperable products to hit the market. The drawback of this approach, however, is that they are infrequent and require engineering teams to travel. Because they typically happen only every couple of months, even a small mistake in an implementation requires that team to delay product release by several. Similarly, such frequent travels might cost too much for small companies wanted to release standards-compliant products. The net result is standards-based products take longer to hit the market, and that consumers are often bound to proprietary products which are often faster and cheaper to create.

The goal of F-Interop is to make conformance and interoperability testing faster and cheaper. It does so by allowing tests to be conducted remotely and online. A server of the Internet plays the role of a "virtual room" in which vendors meet to test their IUTs. The IUT itself does not leave the vendor's premises; instead, an agent running on a computer at the vendor's connects to the server. The agent then remotely drives the IUT and goes through the different test cases. This means that a vendor can launch a conformance testing session at any time, possibly as part of its continuous integration process. Interoperability testing means that different vendors connect to the system at the same time.

3 The F-Interop Platform

Fig. 2 shows the software architecture of F-Interop. The architecture is 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. The following sub-sections describe the different blocks in Fig. 2.

3.1 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 the RabbitMQ[2] as the underlying message-passing mechanism. It acts as a secure message broker between all the components through encrypted channels.

[2] `https://www.rabbitmq.com/`

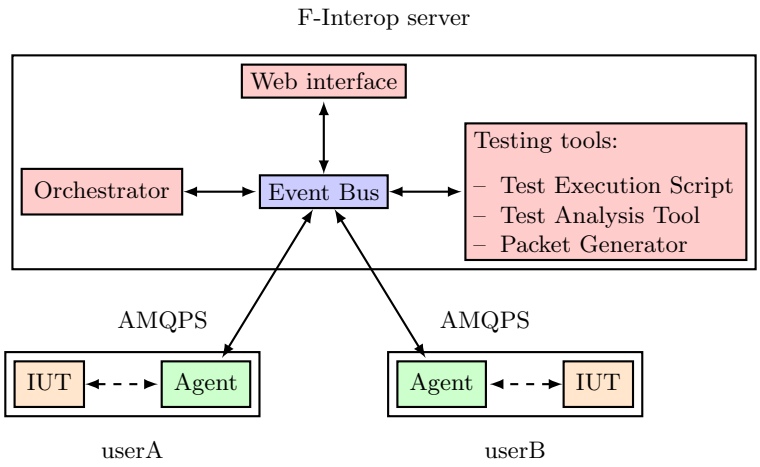

Fig. 2. The F-Interop architecture.

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. 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. Data plane messages contains the raw data exchanged between the IUTs.

Vendors conducts interoperability tests in virtual independent "rooms". We use the virtual host mechanism of RabbitMQ to ensure isolation between concurrent rooms.

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.

3.2 Agent: Connecting Users to the Platform

The "agent" is a program a user downloads from the F-Interop website, and which allows him/her to connect an IUT to the F-Interop server. Communication between the agent and the server is authenticated and secure. Through the agent, the F-Interop server can (remotely) interact with the IUT, for example by changing configuration or injecting packets. Similarly, the agent reports events to the server, such as sniffed packets.

3.3 The Orchestrator

The orchestrator plays a purely administrative role: it monitors the users that are connected, activates the rooms currently in use and starts/stops the test sessions. It is also in charge of provisioning the message broker and updating

firewall rules when test sessions are activated. It does so by spawning/killing the processes of the different components connected to the event bus. It uses the `supervisor` process control system[3].

3.4 Test Session

A test session can be started once the different users are connected and the necessary components are provisioned by the orchestrator. The role of the test session is to generate verdicts that corresponds to test cases. A test session corresponds to one test description. While the F-Interop platform does not impose a particular organization of a test session (i.e. it operates as a black box which generates test verdicts), it is typically composed of a test execution script, a test analysis tool and (optionally) a packet generator.

Test Execution Script. The test execution script (TExS) is the code that describes the configurations and the steps of each test case. It is a translation of a test case of the test descriptions (TD) into machine understandable language. Just like the TDs, the TExS describes the set of steps that need to be executed. Typically, there are 3 types of steps:

- STIMULI: an action for stimulating the IUT (e.g. sending a CoAP message).
- CHECK: the action of validating the communication (e.g. check that the field X is equal to value Y).
- VERIFY: the action of verifying that an IUT behaves correctly (e.g. verify that resource A updated its value to B).

Test Analysis Tool. The Test Analysis Tool (TAT) is the component that performs the verification of traces during a test session. F-Interop provides TATs for different protocols, which run after the message exchange is finished. The TAT issues three types of verdicts: PASS when test purpose of the test case is verified, FAIL when there is at least one fault, INCONCLUSIVE when the behavior of the IUTs does not apply to the one described in the test purpose. The architecture support TATs which perform step-by-step analysis.

TATs are created both by the F-Interop core team and by external contributors. The F-Interop API specification defines the format of the messages a TAT will receive from the Event Bus, and the format of the messages it can produce.

Packet Generator In some conformance tests, a packet generator component can be used to generate packets for the IUT. This component can for example implement the behavior of a CoAP server when the IUT implements a CoAP client. Because it has full control over its packet generator, the F-Interop server can purposely generate wrongly formatted messages to verify the correct behavior of the IUT.

[3] `http://supervisord.org/`

3.5 Web Interface

The F-Interop web interface allows the user to select a test description from a list of available tests, start the execution of the test description and follow the execution of the different test cases. In some cases, the web interface can request the user to take some action (e.g. switch off a node). The web interface also allows the user to retrieve the test report. The web interface communicates with the rest of the system by sending/receiving message over the Event Bus.

4 Example Remote Interoperability Tests

This section shows how the F-Interop architecture is used to execute the CoAP interoperability test from Table 1 between userA and userB. userA has implemented a CoAP server, userB a CoAP client. They want to verify that userB's CoAP client can issue a CoAP `GET` request to userA's CoAP server.

userA and userB agree on a date perform the interoperability testing, and create an account on the F-Interop server. At that date, they download the agent from the F-Interop web site, and connect it to the server using their user credentials. Once connected, the users only interact with the F-Interop web interface.

On the web interface, they create a common room and select the CoAP test description. Because the CoAP implementations of userA and userB are computer programs, the agent of each user creates a virtual `tun` interface. The `tun` interfaces acts as a secure tunnel between userA and userB's agents, which passes through the F-Interop server.

The users then follow the instructions on the web interface: userB issues a CoAP `GET` request to userA's CoAP server. During this exchange, the F-Interop server captures the packets exchanged. The users then indicate the test is over and verify that the exchange behaved correctly; the F-Interop server analyses the packets exchanged and issues a verdict. Fig. 3 shows the web interface.

5 Discussion

F-Interop is an ongoing project. Its architecture is not written in stone and the F-Interop team is always looking to enhance it to be able to handle addition test configurations. This section contains addition features being worked on.

Testbed integration. Several low-power wireless mesh testbeds exist which contain a large number of nodes. The goal of F-Interop is to allow tests to be run on those testbeds, for example by running the user's firmware and a reference firmware side-by-side on different nodes in the testbed. In that context, F-Interop tests could be launched periodically as part of continuous integration.

Accurate end-to-end latency measurement. There is a delay between the user premises and the F-Interop system; in some cases, this delay could code event re-ordering and false verdicts. The F-Interop team is contemplating

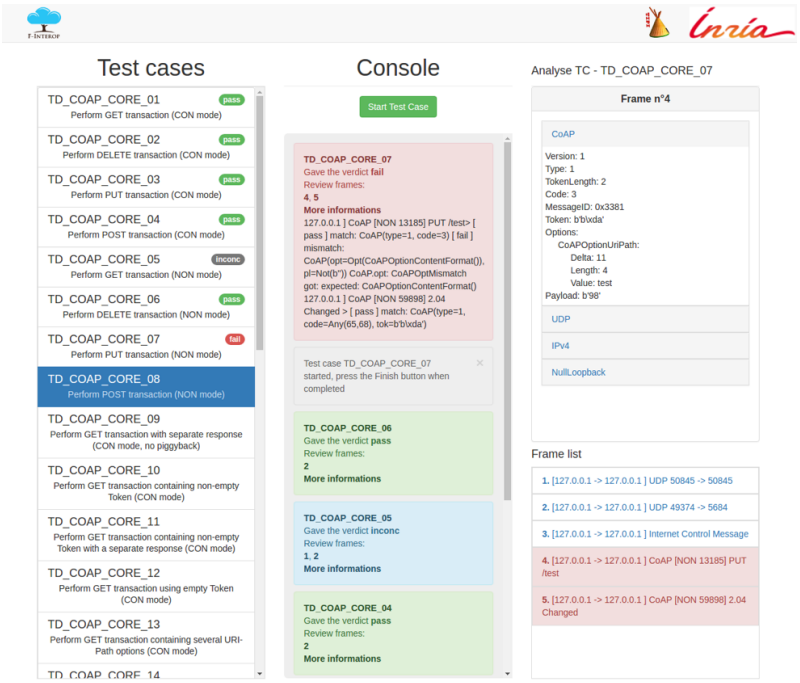

Fig. 3. Web interface after 7 tests have been run.

building a board which the users would use in their premises, and which would synchronize to GPS and timestamp events with a 10-100 ns accuracy.

Energy measurement capabilities at the user. Energy consumption is an important part of any low-power wireless product; some test cases could target energy consumption. A board which would measure the energy consumption of the IUT would enable a large number of addition test cases.

References

1. Carsten Bormann. Test Descriptions for ETSI plugtest CoAP#4. Technical report, ETSI, London, United Kingdom, 7-9 March 2014.
2. Carsten Bormann. 6Lo Test Descriptions, ETSI 6TiSCH/6lo plugtest. Technical report, ETSI, Berlin, Germany, 17-19 July 2016.
3. Maria Rita Palattella, Xavier Vilajosana, Tengfei Chang, and Thomas Watteyne. 6TiSCH Interoperability Test Descriptions for the ETSI 6TiSCH/6lo Plugtests. Technical report, ETSI, Berlin, Germany, 17-19 July 2016.
4. Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP), June 2014.
5. Sebastian Ziegler, Serge Fdida, Thomas Watteyne, and Cesar Viho. F-Interop – Online Conformance, Interoperability and Performance Tests for the IoT. In *International Conference on Interoperability in IoT (InterIoT)*, Paris, France, 26-28 October 2016. EAI, Springer.