F-INTEROP

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

Deliverable D1.1
Requirements specification report

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:	**M6**
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
Editor(s)/Author(s)	**César Viho & Federico Sismondi (INRIA)** Maria Rita Palatella & Luca Lamorte (UL) Eldad Zack (EANTC) Michele Nati (DigiCatapult) Thomas Watteyne & Rémy Léone (INRIA) Miguel Angel Reina Ortega (ETSI)
Version	2.1
Keywords	Remote testing, online testing, F-Interop User's needs, F-Interop Contributor's needs, F-Interop-Platform requirements, Interoperability and conformance testing, performance testing, scalability testing, resiliency testing, security and privacy risk testing, Internet of Things (IoT), CoAP, 6TiSCH, OneM2M

Abstract

This deliverable, **D1.1 – Requirements specification report,** is the first version of the report that describes key requirements for a generic F-Interop testing architecture that eases online interoperability and performance testing.
To this purpose the ***F-Interop-Platform*** to be built is defined and is initially composed of the three testbeds (IoT-Lab, Fed4FIRE, OneLab) and first version of testing tools provided by the F-Interop project partners. A ***F-Interop-User*** is any entity that wants to use the F-Interop-Platform to test its IoT device, system or application. A ***F-Interop-Contributor*** is any entity that provides additional testing tools or resources into the F-Interop-Platform.
Three emerging IoT technologies (6TiSCH, CoAP and oneM2M) have been selected to cover all layers of the IoT protocol stack. Based on the test scenarios that have been used during recent face-to-face interoperability and performance testing events, key components as well as F-Interop-User and F-Interop-Contributor needs for running online remote testing have been identified. The key requirements of the F-Interop-Platform are then derived and synthesized in shape of a table.

Table of Contents

List of Figures

List of Tables

List of Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
ATS	Abstract Test Suite
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments
ETS	Executable Test Suite
ETSI	European Telecommunications Standards Institute
EU	European Union
F2F	Face-to-face
FI	F-Interop
FI-CN	F-Interop Contributor Need
FI-PR	F-Interop Platform Requirement
FI-UN	F-Interop User Need
GD	Golden Device
HTTP	Hypertext Trasnfer Protocol
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IoP	Interoperability
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Standards Organization
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunication Union
IUT	Implementation Under Test
KPI	Key Performance Indicator
LLN	Low-power and Lossy Network
M2M	Machine to Machine
MAC	Media Access Control
MOS	Mean Opinion Score
NBI	Northbound Interface
NFV	Network Function Virutalisation
OS	Operating System
PIXIT	Platform Implementation eXtra Information for Testing
PoC	Proof of Concept
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request For Comments
RPL	(IPv6) Routing Protocol for LLNs
R&D	Research & Development
SBI	Southbound Interface
SDN	Software Defined Networking
SME	Small Medium Enterprise

SSL	Secure Sockets Layer
SUT	System Under Test
TBaaS	Test Bed as a Service
TCP	Transmission Control Protocol
TD	Test Description
TDMA	Time Division Multiple Access
TLS	Transport Layer Security
TSCH	Timeslotted Channel Hopping
TTA	Telecommunications Technology Association
UK	United Kingdoms
UDP	User Datagram Protocol
URL	Uniform Resource Locator
US	United States
VNF	Virtual Network Function
VoIP	Voice over Internet Protocol
WP	Work Package
WPL	Work Package Leader
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1 Introduction

1.1 About F-Interop

F-Interop is a Horizon 2020 European Research project, which proposes to extend the European research infrastructure (FIRE+) with online and remote interoperability (IoP) and performance test tools supporting emerging technologies from research to standardization and to market launch. The outcome will be a set of tools enabling:

- Standardization communities to save time and resources, to be more inclusive with partners who cannot afford travelling, and to accelerate standardization processes;
- SMEs and companies to develop standards-based interoperable products with a shorter time-to-market and significantly lowered engineering and financial overhead.

F-Interop intends to position FIRE+ as an accelerator for new standards and innovations.

1.2 Deliverable Objectives

1.2.1 Work package Objectives

The overall objectives of this work package **WP1 – "Requirements and Architecture design"** are to:

- Analyze and specify the online testing tools requirements,
- Analyze and specify personal data protection and security requirements,
- Design and specify the F-Interop architecture

The work package WP1 is composed of the *three tasks described below.*

1.2.2 Task Objectives

1.2.2.1 Task T1.1 - Testing tools requirements analysis

IoT deals with several domains coming from applications, via middleware to infrastructures used by the IoT components/objects to communicate with each other. For each domain, there are different approaches for testing that have similarities but some specific requirements. By studying those testing approaches (including platforms, test beds, test events, testing methodologies and tools, etc.), we will gain good knowledge about main issues and requirements and we will be capable of identifying best practices for remote interoperability and performance testing. The purpose of this task T.1.1 is to provide clear ideas on those requirements so that it helps in defining a generic F-Interop testing architecture that eases remote, interoperability and performance testing.

1.2.2.2 Task T1.2 – Privacy and security by design

This task will investigate how to provide security and privacy by design to the F-Interop architecture, when federating different test beds spread in different locations/countries and owned by different entities. First of all, it will be needed to combine the different privacy/security policies implemented in each of them. The requirements for ensuring: a) privacy of the test results, b) confidentiality of the experiment (i.e., privacy of the exchanged data, and scripts deployed on the shared

test beds) will be defined. This task T1.2 will specify security and privacy requirements of the F-Interop architecture.

1.2.2.3 Task T1.3 - Architecture and FIRE+ integration design

Based on the initial architecture described in the F-Interop proposal, the requirements studied in T1.1 and T1.2, the requirements expressed towards a Testbed as a Service (TaaS) model, T1.3 will design the architecture for the F-Interop platform. The basic architecture will be ready, while updates will be made based on the lessons learned throughout the project, open calls and feedback from the users. This task T1.3 will provide clear and detailed architecture specification to be used by the subsequent work packages.

1.2.3 Deliverable Objectives and Methodology

1.2.3.1 Deliverable Objectives

This deliverable **D1.1 – Requirements specification report** is the first version of the report that describes requirements for a generic F-Interop testing architecture that eases online interoperability and performance testing.

1.2.3.2 Methodology

As a **first step, we have** defined the ***F-Interop-Platform***. It is composed of the following three testbeds: IoT-Lab, Fed4FIRE and OneLab brought by partners into the F-Interop project as well as testing tools and IoT devices provided by partners of the F-Interop project. This ***F-Interop-Platform*** will be extended with additive testing tools, devices and testbeds.
As a **second step we have** identified different types of actors of the ***F-Interop-Platform***. We have distinguished two main types of participants. In the following we will call them: ***F-Interop-Contributor*** and ***F-Interop-User***.

- An ***F-Interop-Contributor*** (FI-Contributor) is any entity that provides testing tools into the F-Interop-Platform as well as testbeds and devices that are added to extend the existing testbeds.
- An ***F-Interop-User*** (FI-User) is any entity that has an IoT device, system or application to be tested (called IUT, Implementation Under Test) and wants to use the F-Interop-Platform to test it.

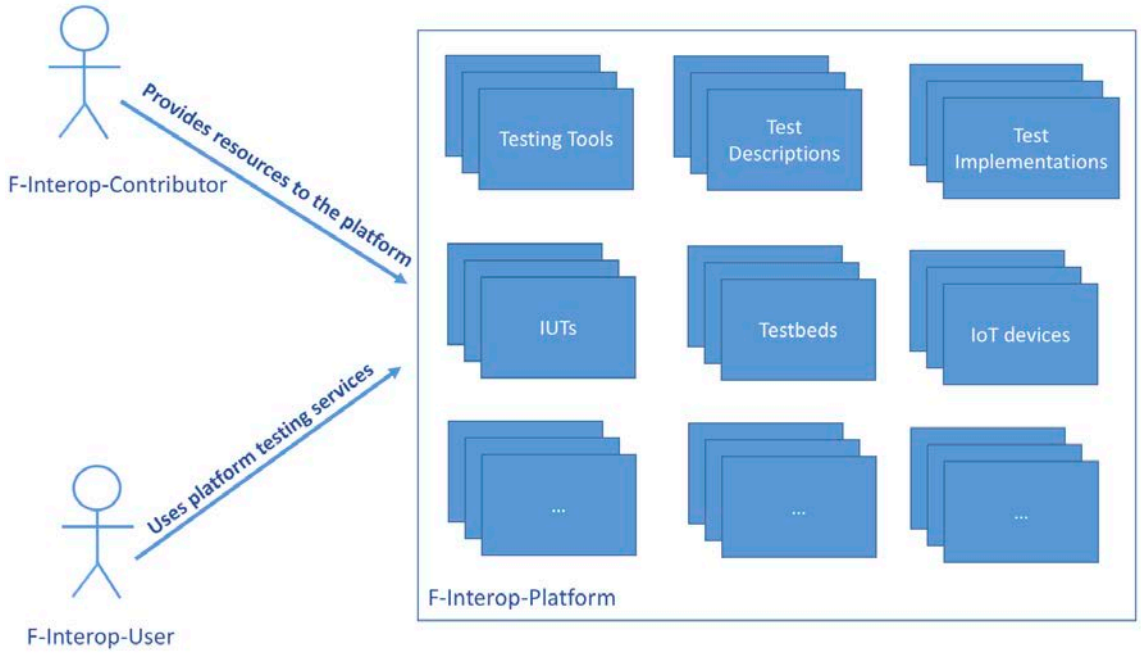


Figure 1 - F-Interop main components

An entity can be both *F-Interop-Contributor* and *F-Interop-User*. An *F-Interop-User* can become a *F-Interop-Contributor* as soon as it makes his/her IUT available for other *F-Interop-Users* in the *F-Interop-Platform.* **Partners of the F-Interop project are the main and first F-Interop-Contributors**.

As a **third step**, we have considered the two types of testing tools that the F-Interop has to deal with: ***Online Interoperability testing*** and ***Online performance testing***. We decided to select some of the targeted emerging IoT technologies that cover as many layers/aspects as possible of the IoT protocol stack. After internal discussion, we decided to focus first on the following protocols: 6TiSCH, CoAP and oneM2M. For each of these two types of testing and the selected protocols, we have investigated the state of the art (existing methods and tools) for testing, and we have studied and compared existing IoT related testing solutions and tools. These studies helped us starting the discussion on what a user might expect from the F-Interop-Platform.

Based on the test scenarios that have been developed and used during previous and recent interoperability face-to-face (F2F) interoperability testing events, we started studying what is needed for doing the same but in an online and remote manner. This work helped us in identifying key components for online remote testing, as well as F-Interop-User and F-Interop-Contributor needs for running online remote testing:

- The **F-Interop-User needs** (FI-UN) identify features or functionalities expected by a user or other stakeholder that act from a perspective external to the F-Interop-Platform. They will contribute to the definition of F-Interop-Platform requirements.
- The **F-Interop-Contributor needs** (FI-CN) identify features, functionalities or interfaces expected by a stakeholder that wants to provide contributions in the form of testing tools/resources or IoT systems/applications into the F-Interop-Platform. They will contribute to the definition of F-Interop-Platform requirements.

The following sections describe the first identified needs. Based on these needs and using the methodology introduced in section 1.2.3.3 below, the **forth step** was to define the set of F-Interop-Platform requirements presented in section 4:

- The **F-Interop-Platform requirement** (FI-PR) express desirable features, functionalities and properties that when implemented will lead to the achievement of at least one F-Interop-User need and/or one F-Interop-Contributor need.

1.2.3.3 Methodology for deriving F-Interop-Platform Requirements

Figure 2 illustrates the process flow for defining F-Interop-Platform requirements, from the collecting phase of F-Interop-Contributor and F-Interop-User needs to the deriving of the final F-Interop-Platform requirements specification.

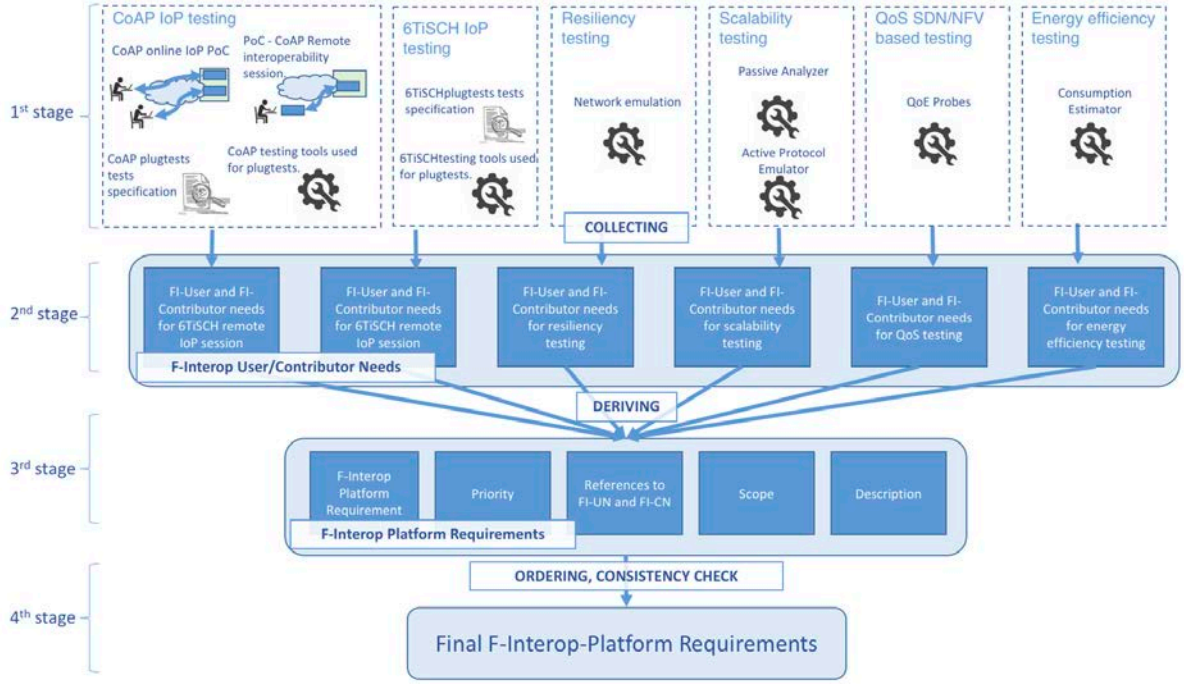


Figure 2 - Methodology for deriving F-Interop-Platform requirements.

The diagram shows the different stages of the process. At the 1st stage different groups of inputs corresponding to the different usages and functionalities the F-Interop-Platform aims to provide. Based on these categories we generated a first set of needs (FI-UNs and FI-CNs). These needs form an extensive coverage of all the services that are needed from F-Interop-Platform to achieve online interoperability and performance testing for the targeted protocols. Among these needs we expected to find several overlaps in the functionalities or properties that are demanded, since this first collection is not intended to be optimal, but exhaustive as possible. These first gathered needs are statements written in natural language accompanied with diagrams when needed to ensure clarity. These statements aim to be verifiable, clear, unambiguous and implementation-agnostic.

In the 2nd stage we put all these needs together trying to avoid repetition of functionalities. They are presented in the form of two tables, one for FI-UNs and another for FI-CNs. In this phase a unique identifier is assigned to each need.

In the 3rd stage we derived the FI-Platform requirements (FI-PR) from the FI-UN and FI-CN. FI-PRs define what the developers in the F-Interop-Platform should implement. Each FI-PR is composed of five of elements: (1) a unique identifier, (2) Field/Step, (3) the requirement statement, describing a FI-Platform function, a service provided or an operational constraint, (4) a detailed description of the requirement and (5) one or more references to FI-UN and/or FI-CN, providing a traceable reference to a user/contributor-like need that allows to have verifiable and user-oriented description of the implemented functionality.

The 4th stage fourth level of the diagram was the last phase of this study. It consisted mainly in a final evaluation of the FI-PRs, focusing in ordering and verifying their consistency. In here, an initial strategy for the implementation of these FI-PRs was provided.

2 Requirements for online interoperability and conformance testing

Conformance testing aims at verifying that an IUT behaves as foreseen in its specification. Interoperability testing aims at verifying that multi-vendor and multi-technologies interconnected components and/or applications interact correctly. Currently interoperability and conformance testing are done F2F. This section is dedicated to the identification of requirements for online remote interoperability and conformance testing.

2.1 About interoperability testing

2.1.1 Main architectures and approaches for interoperability testing

Interoperability testing concerns interactions between at least two IUTs. There are two main interoperability testing architectures: (i) the multiparty interoperability testing architecture where there are at least three IUTs to be tested (see Figure 3), and (ii) the one-to-one interoperability testing architecture which deals with the interaction between two IUTs (see Figure 4). We will consider the one-to-one interoperability testing architecture since it is currently the most common.

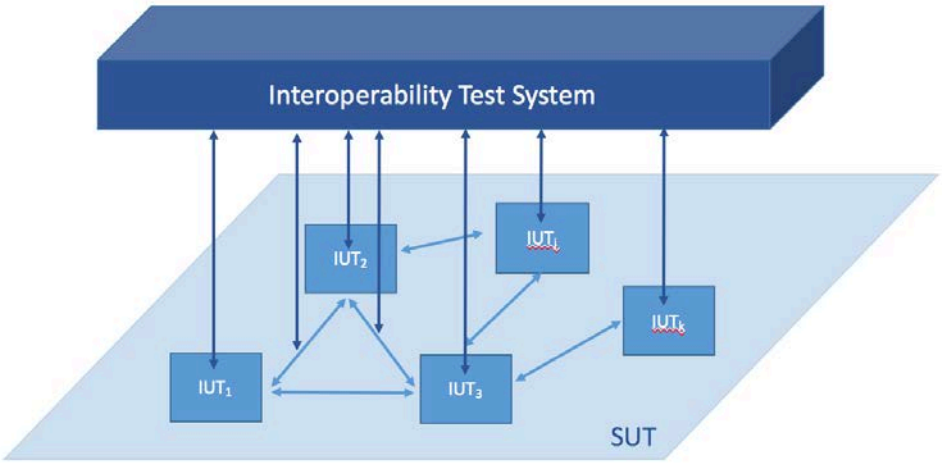
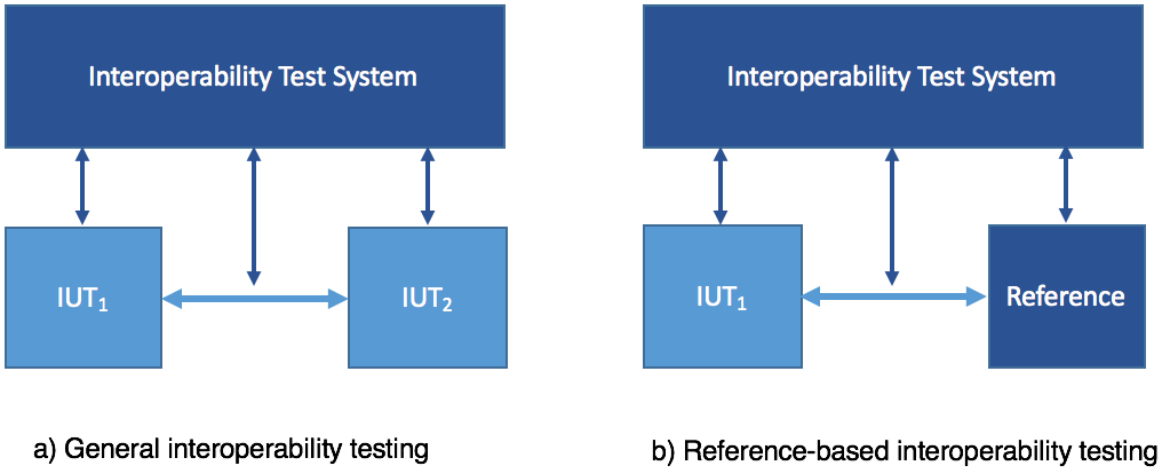


Figure 3 - Multiparty interoperability testing architecture

In the specific case of the one-to-one interoperability testing architecture, there are two main ways to realize interoperability testing: First, the standard interoperability testing, where two different IUT of a protocol are tested against each other (see Figure 4.a). And second, the so-called reference-based interoperability testing (see Figure 4.b), where one of the two IUTs is considered as a reference implementation and the other one (called the target IUT) is tested against the reference IUT.



a) General interoperability testing

b) Reference-based interoperability testing

Figure 4 - One-to-one interoperability testing architecture

Regarding the interaction between the test system and the IUTs, there are two main approaches for interoperability testing:

1) The **active approach** of testing where the test system controls all the actions to be done by the IUTs by alternating stimuli and observations
2) The **passive approach** that minimizes the control of the IUTs and bases the emitted verdicts on the observed interactions.

2.1.2 The main steps in conformance and interoperability testing

Interoperability testing follows similar main steps as in conformance testing. In the following sections we provide the main steps in conformance testing and after that we indicate steps that apply also for interoperability testing while indicating steps that are specific to interoperability testing.

2.1.2.1 Mains steps in conformance testing

The **first step** in conformance testing is to specify the conformance test purposes/objectives. A test purpose defines informally one of the functionalities/properties to be tested. For each test purpose, a scenario description is provided containing different steps that help testing the associated test purpose. This is generally done in shape of documents (an example is given in annex 7.1). Each test description generally contains:

1. The test configuration which describes the considered interconnection topology of the IUT and the test system and setting information describing the resources that are required;
2. The test purpose describes the specific property or functionality targeted with this test description/scenario;
3. The test sequence that corresponds to testing actions and steps - stimulus sending, message checking and specific field values to be verified according to the test purpose.

Depending on the context, these test scenarios or test descriptions are translated into abstract test cases, using a standardized language (e.g., TTCN-3) or any other test-

oriented language. They are abstract in the sense that they do not depend on a specific environment or platform. The list of these test cases forms the abstract test suite (ATS).

The **second step** is then to translate the ATS into an actual *Executable Test Suite* (ETS) using particular *Platform Implementation eXtra Information for Testing* (PIXIT) that provides actual interfaces, addresses, actual timers' values, etc., to be considered during the test execution.

The **final step** is to execute the ETS against the IUTs, to emit corresponding verdicts (PASS, FAIL or INCONCLUSIVE) and to provide testing results. This last step is complex, as it needs coordination between different business entities and/or experimenters.

2.1.2.2 Main steps in interoperability testing

As discussed above, interoperability testing follows similar main steps as conformance testing. The main difference is, depending on the context, that it may be impossible to have an executable test suite. It means that the test execution is done directly based on the test descriptions provided in the first step together with a particular PIXIT that provides actual interfaces, addresses, actual timers' values, etc., to be considered during the interoperability test execution.

In the case of interoperability testing, the test execution phases are often much more complicated as there are at least 3 parties involved: the test authority (or test lab) and at least two vendors. In some cases, there will also be a customer/end-user involved. In the framework of F-Interop, the test authority is represented by the F-Interop-Platform, while the vendors/customers/end users are represented by the F-Interop-Users. F-Interop-Contributor may also be considered a part of the test authority, when it contributes with tools, test descriptions, ATS or complete ETS's.

2.1.3 Towards key requirements identification for remote interoperability testing

As stated in the introduction (see Section 1.2.3.2), we have chosen two IoT-related protocols to study the needs of remote interoperability testing. These two protocols are 6TiSCH and CoAP. The 6TiSCH protocol suite includes IEEE802.15.4e TSCH, RPL and 6LoWPAN, which are rather lower layer protocols. When performing 6TiSCH interoperability tests, one needs to check features at the MAC layer, as well as RPL-related, or can test new 6TiSCH protocols (like 6top). By considering 6TiSCH, it will help us come up with better ideas for similar lower layer protocols. In contrast to 6TiSCH, CoAP is an application layer protocol as it runs over the transport layer. Through the study of these two protocols we will be able to cover the requirements for interoperability and conformance testing of lower layer protocols as well as upper layer protocols. These studies are done for each of the three steps (see Section 2.1.2). For each step, needs for the F-Interop-User, the needs for F-Interop-Contributor and requirements for the F-Interop-Platform are provided.

2.2 The case of CoAP interoperability testing

2.2.1 CoAP protocol: short overview

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. It is a simple Client/Server protocol with messaging and request/response as just features of the CoAP header. The CoAP messaging model is based on the exchange of messages over UDP between endpoints.
CoAP was designed with the following main features: (1) constrained RESTful protocol fulfilling M2M requirements, (2) stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP, (3) UDP binding with optional reliable unicast and best-effort multicast support, (4) asynchronous transaction support, (5) low header overhead and parsing complexity, (6) URI and Content-type support and (7) built-in resource discovery.

The following image shows a typical client's GET request with its corresponding server's response:

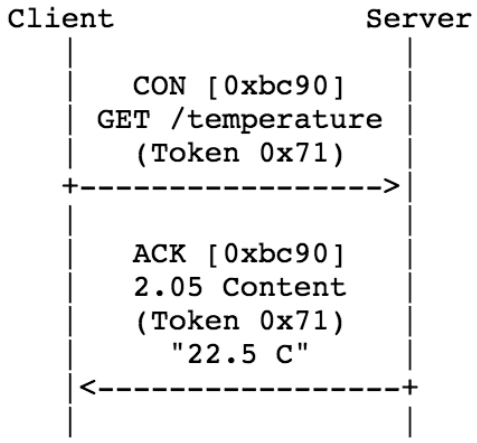


Figure 5 - CoAP message exchange example.

The CoAP protocol is specified by four main documents:

- RFC-7252 [2] specifying the base protocol,
- RFC-7641 [3] (observing resources) extends the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server,
- draft-ietf-core-block-19 [4] (block-wise transfer; work in progress) extends basic CoAP with a pair of "Block" options, for transferring multiple blocks of information from a resource representation in multiple request-response pairs, and
- RFC-6690 [5] (link format) specifies a link format for use in Constrained RESTful Environments (CoRE) Resource Discovery.

Interoperability testing for CoAP implies verifying that messages exchanged between a CoAP client implementation and a CoAP server implementation complies with these four above-mentioned specifications.

2.2.2 State of the art on CoAP interoperability testing

Most of the existing interoperability testing activities is done face-to-face (F2F) in a testing lab, in the premises of the entities that develop the IUTs or during the so-called interoperability events such as the CoAP Plugtests organized by ETSI. The first ETSI CoAP Plugtest was co-located with the 83rd IETF meeting in Paris 24 - 25 March 2012; the second was in Nice (France), 27-29 November 2012 and the third was held in Las Vegas, USA, from 19 to 22 November 2013.

Interoperability tests were run on first implementations of CoAP servers and clients. Features tested included the base CoAP specification, CoAP Block Transfer, CoAP Observation and the CoRE Link Format.

During these events participants interconnect their devices and go through the execution of a series of predefined tests (see an example in Annex 7.2). While tests are being executed, participants must analyze the messages exchanged between the IUTs and verify they are compliant with the test specification. More than 3000 interoperability tests were conducted. These activities are extremely time consuming and error-prone. Moreover, fixing bugs and other problems that might appear during the test execution introduce delays in the tight schedules of the events.

The following sections describe the desired F-Interop features and properties, in the form of needs, seeking to minimize the aforementioned problems.

2.2.3 FI-User and FI-Contributor needs for CoAP online interoperability testing

This section synthesizes the first lessons learnt from studying what is required when trying to move from the usual/classical F2F CoAP interoperability testing to online CoAP interoperability testing.

2.2.3.1 FI-User and FI-Contributor needs for CoAP remote interoperability testing

Two proof of concepts (PoCs) were deployed to help with the requirements gathering phase; (a) online interoperability session between two F-Interop-Users, and (b) interoperability session between a FI-User's remote IUT against automated IUT deployed in the F-Interop-Platform.

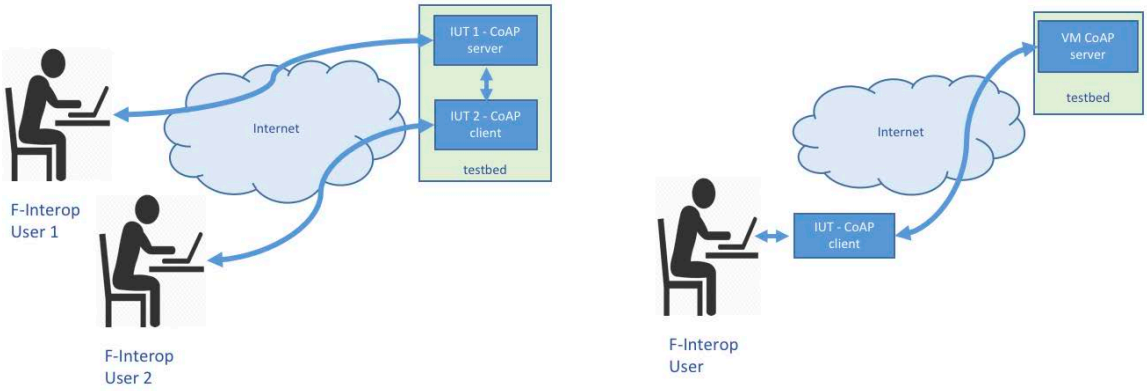


Figure 6 - PoCs used for gathering requirements

From the PoCs on **remote interoperability session** the following F-Interop-User and F-Interop-Contributor needs for CoAP were gathered:

- FI-User needs a communication channel/mechanism with other FI-Users participating in the interoperability (IoP) session for coordination purposes.
- FI-User should have a simple way of deploying its implementation in the FI-Platform for running an interoperability session.
- FI-Contributor should have means to upload its implementation to FI-Platform for the usage by other FI-Users.
- FI-User needs to be able to interact with its implementation when the user deploys it in one of the testbeds of the FI-Platform.
- FI-User should be provided with a visualizing tool for analysing the interaction between involved entities while during the test execution.
- FI-User needs to select which test suites / test cases to run on the FI-Platform.
- FI-User may be provided with a CoAP-visualizing tool for showing test results verdicts while executing the tests.
- FI-User needs to be able to select which implementation it wants to use in the interoperability session (for the automated IUT hosted by FI-Platform).
- FI-User needs to be provided with tools that help it overcome typical reachability problems (like filtered UDP traffic on firewalls).
- FI-User should have an interface displaying information regarding the state of the automated execution of tests.
- FI-User may be able to control the automated execution of the tests (i.e., the user may issue commands such as stop a test case, skip a particular test case, restart the test suite, etc.).
- FI-User needs to be able to launch the execution of the tests and abort when needed.
- FI-User should be provided with a tool describing the actions needed from user's side (example "execute test TD_COAP_CORE_02").

2.2.3.2 FI-User and FI-Contributor needs for abstract tests scenario description and specification

- FI-User needs to be able to access/view the list of existing CoAP tests, including those for previously defined versions, e.g. the user might want to see the test scenario description and specification for TD-CoAP-OBS-rfc7641, TD-draft-ietf-core-observe-16 and/or TD-draft-ietf-core-observe-15).
- FI-Contributor needs to be able to access, add, edit, modify and delete CoAP tests.
- FI-Contributor needs to be provided with templates/models when defining new test scenario description or specification for CoAP.

2.2.3.3 FI-User and FI-Contributor needs for providing executable test

- FI-Contributor should be able to view and suggest modifications to the CoAP testing tool source code.
- FI-Contributor should be able to access, add, edit, modify and delete CoAP tests implementation files for CoAP testing tool.
- FI-Contributor should have the option to develop tests implementations files in different programming languages.
- FI-User needs to have means for providing PIXIT (Protocol Implementation eXtra Information for Testing).
- FI-Contributor may have means of providing CoDeC (Coding and Decoding) and libraries for the test execution.
- FI-Contributor needs to be able to upload an IUT for remote automated execution.
- FI-Contributor should be able to upload his/her own testing tools to the FI-Platform.

2.2.3.4 FI-User and FI-Contributor needs for test execution

- FI-User needs to be able to define in the F-Interop-Platform which tests to run when executing automated with the IUT remotely on the F-Interop-Platform.
- FI-User may be provided with a CoAP-visualizing tool for showing test results and verdicts.

2.3 The case of 6TiSCH interoperability testing

2.3.1 6TiSCH protocols suite: a short overview

The IEEE802.15.4e "Timeslotted Channel Hopping (TSCH)" is a 2012 amendment to the Medium Access Control (MAC) portion of the IEEE802.15.4 standard. TSCH is the emerging standard for industrial automation and process control in low-power wireless, with a direct inheritance from WirelessHART and ISA100.11a. Defining IPv6 over TSCH, 6TiSCH is a key to enable the further adoption of IPv6 in industrial standards and the convergence of Operational Technology (OT) with Information

Technology (IT). Details about the activities of the 6TiSCH working group can be found in its charter [6].

The nodes in an IEEE802.15.4e TSCH network communicate by following a Time Division Multiple Access (TDMA) schedule. A timeslot in this schedule provides a unit of bandwidth that is allocated for communication between neighbor nodes. The allocation can be programmed such that the predictable transmission pattern matches the traffic requirements of the applications running on the network. This avoids idle listening, and extends battery lifetime for constrained nodes. Channel-hopping improves reliability in the presence of narrow-band interference and multi-path fading.
These techniques enable a new range of use cases for Low-power and Lossy Networks (LLNs), including:

- Control loops in a wireless process control network, in which high reliability and a fully deterministic behaviour are required.
- Service Provider networks transporting data from different independent clients, and for which an operator needs flow isolation and traffic shaping.
- Networks comprising energy harvesting nodes, which require an extremely low and predictable average power consumption.

IEEE802.15.4e only defines the link-layer mechanisms. It does not define how the network communication schedule is built and matched to the traffic requirements of the network.

2.3.2 State of the art on 6TiSCH interoperability testing

Up until now, 6TiSCH interoperability testing activities have been conducted through F2F interoperability Plugtests events. Two of these events have been organized by ETSI, in July 2015 in Prague, CZ, and in February 2016 in Paris, France.

During those 6TiSCH Plugtests, the organizers created a Golden Device (GD), which is a pre-programmed low-power wireless device running firmware known to correctly implement the 6TiSCH standards being tested. This GD acts as a reference implementation of 6TiSCH that corresponds to the reference IUT in the F-Interop architecture (see Figure 4). Besides running 6TiSCH, this GD can be used as a test proxy that injects frames, or as a test observer that observes what the IUT is sending and receiving. In that case, it serves as a tool for simplifying testing, not as a reference implementation. While the tests are being executed, developers manually analyze the messages exchanged between the IUTs (using sniffer like wireshark), and verify they are compliant with the test specification. The manual nature of executing the tests is time-consuming and error-prone. Moreover, fixing bugs that might appear during the test execution introduce delays in the tight schedule of the two or three days duration of the Plugtests.

The following sections describe the desired F-Interop features and properties, in the form of requirements/needs, seeking to minimize the aforementioned problems.

2.3.3 FI-User and FI-Contributor needs for 6TiSCH online interoperability testing

6TiSCH is different from application-level protocols such as CoAP in that it is sensitive to link-layer delays. 6TiSCH nodes implement the IEEE802.15.4 TSCH link-layer standard. In a 6TiSCH network, neighbor nodes are typically de-synchronized by at most 100's of µs.
Communication delays between the user premises and the F-Interop server are typically in the order of 100's of ms, 3 orders of magnitude higher than the allowed de-synchronization within the 6TiSCH network.
As a result, time-sensitive portions of testing 6TiSCH (e.g. sending a link-layer acknowledgment) cannot involve "real-time" interaction between the user and the F-Interop server. The F-Interop solution MUST allow for latencies typical for an Internet connection between the user premises and the F-Interop server. Although the exact solution depends on the architecture to be developed, this may mean having a device at the user premises which takes care of low-level, time-sensitive aspects of 6TiSCH and is driven by the F-Interop server which issues high-level commands to it.

2.3.3.1 FI-User and FI-Contributor needs for tests scenario description and specification

- FI-User needs to be able to access the list of existing 6TiSCH test specifications, including previously defined versions.
- FI-Contributor needs to be able to access, add, edit, modify and delete 6TiSCH test specifications.
- FI-Contributor should be provided with templates/models when defining a new test scenario description or specification for 6TiSCH.
- FI-User needs to be provided with some visual input (e.g. a popup window on the web interface, a line on a command line terminal, etc) describing the actions he needs to do (e.g. “start executing TD_6TiSCH_SYN_01”).
- FI-User needs to be able to launch the execution of a particular test suite, and abort when needed.
- FI-User may be able to control the flow of execution of a particular test suite (e.g. skip particular tests, restart the test suite).
- FI-User should be able to follow the execution of the test suite, through some interface (e.g. web page dynamically updating as the tests pass/fail).
- FI-User should be provided with tools that help diagnose connectivity issues between the user's premises and the F-Interop-Platform.
- When testing interoperability between the user's implementation and a known-to-work implementation, the FI-User needs to be able to select which implementation to test against.
- FI-User may be provided with tools to analyse the traffic being exchanged between IUTs, either in real time or offline.
- When a FI-User's IUT is running on a testbed, the FI-User may have access to tools to interact with his IUT as it runs.
- When running an IUT on a testbed, the user MUST have access to the appropriate tools to load its IUT on the testbed in an automated/scriptable way.

2.3.3.2 FI-User and FI-Contributor needs for executable test deriving

- FI-Contributor needs to be able to log into the system as a user and execute test specifications, including the ones he is working on, like any regular user.
- FI-Contributor may be able to view and suggest modifications to source code of the 6TiSCH test specifications.
- FI-Contributor needs to be able to contribute new tests to the 6TiSCH tests suites.
- FI-Contributor should be able to suggest modifications to already existing 6TiSCH test suites.
- Contributor should have the option to develop tests implementations files in different programming languages.

2.3.3.3 FI-User and FI-Contributor needs for test execution

- FI-User needs to be able to tell the F-Interop platform which tests to run when executing an automated test.
- FI-User may be provided with a tool to follow the test execution.

2.4 The case of oneM2M interoperability testing

2.4.1 OneM2M: short overview

oneM2M is a global standards initiative established through an alliance of main worldwide standards organizations for Machine to Machine (M2M) communications and the Internet of Things (IoT). It aims at developing and technical specifications to address the need of a common M2M service layer, which can be easily embedded within various hardware and software to connect with M2M application servers worldwide. These technical specifications cover several aspects such as use cases and requirements for common service layer capabilities, detailed service architecture, protocol and APIs based on the architecture, management aspects, and interoperability and conformance testing.

2.4.2 State of the art on oneM2M interoperability testing

The first interoperability-testing event for oneM2M was organized by ETSI in collaboration with TTA (Telecommunications Technology Association)-Korea in September 2015.
The oneM2M partnership project, particularly as part of the Testing (TST) Working Group (WG) developed the Interoperability Test Descriptions (oneM2M TS-0013) specification that defines what and how different features need to be tested. Those test descriptions were adapted to the three different bindings (HTTP, COAP and MQTT) that oneM2M is considering and defining.
During the interoperability event vendors interconnect their implementations and run the tests by following the procedure defined in the test descriptions.
The first oneM2M interoperability testing event focused on basic functionality and aimed at making vendors test against as many other vendors as possible so that base specifications could be validated and assessed with regards to interoperability.
Further oneM2M interoperability testing events are planned and F-Interop is foreseen as contributing to those by providing online tools to facilitate the development of the implementations as well as the interoperability assessment of the base specifications

and by contributing to the development of the Interoperability Test Descriptions specification.
The following sections describe the desired F-Interop features and properties, in the form of requirements, permitting to carry out the above activities.

2.4.3 FI-User and FI-Contributor needs for oneM2M online interoperability testing

This section describes the needs from users and contributors and the services that F-Interop-Platform has to provide in order to be able to perform online interoperability testing for oneM2M.

2.4.3.1 FI-User and FI-Contributor needs for oneM2M remote interoperability testing

Two test scenarios were taking into account to help with the requirements gathering phase; (a) online interoperability session between F-Interop Users, and (b) online interoperability session against a oneM2M reference implementation.

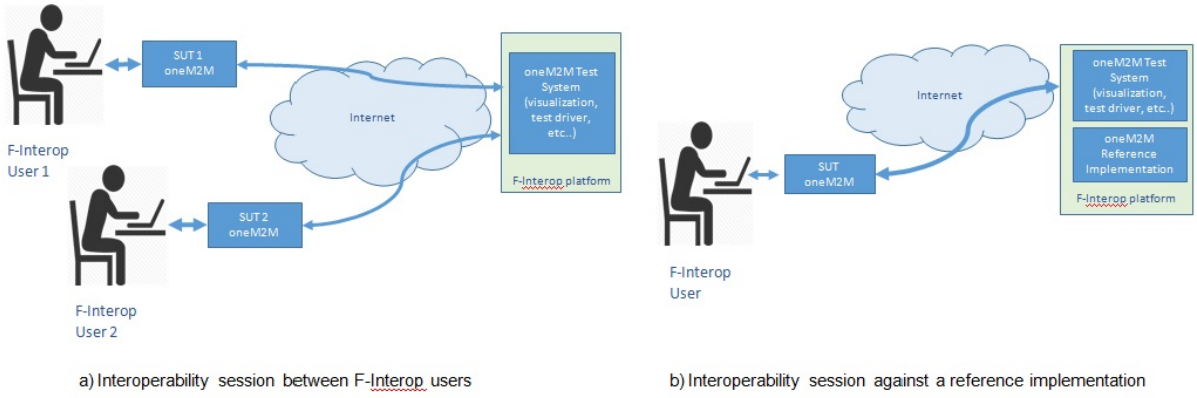


Figure 6 – Test scenarios used for gathering requirements

From the FI-User point of view, the following services need to be satisfied in order to permit an online interoperability session:

- FI-User needs to have a secured communication channel with other FI-Users participating in the IoP session for coordination purposes via FI-Platform.
- FI-Contributor needs to have means to upload, edit, remove and share its implementation to FI-Platform with other FI-Users.
- FI-User should be provided with a tool that visualizes the traffic in real-time.
- FI-Contributor needs to be able to upload other test tools (i.e. Wireshark dissector) to the FI-Platform

2.4.3.2 FI-User and FI-Contributor needs for test descriptions specification

- FI-User needs to have access to the existing oneM2M test descriptions, including old versions.

- FI-Contributor needs to be able to add, edit, modify and delete oneM2M tests.
- FI-Contributor may be provided with templates when defining new test description for oneM2M.

2.4.3.3 FI-User and FI-Contributor needs for automated interoperability test execution

- FI-User needs to be provided with a tool that visualizes the available tests in the F-Interop platform.
- FI-User may be provided with a visualizing tool for showing test results verdicts.
- FI-User needs to be able to select the tests to be run. The selection may be done manually or automatically after providing information about the implementation's supported features.
- FI-User needs to be able to select which reference implementation (for the reference implementation scenario) or other user's implementation (for the user-to-user case) to use in the interoperability session.
- FI-User needs to be able to control the execution of the tests (i.e., the user may issue commands such as stop a test, skip a particular test, restart a test campaign, etc.) for the reference implementation scenario.
- FI-User needs to have means to get notified in real time on the actions to be performed for the correct execution of the test.
- FI-User should be able to view and suggest modifications to the interoperability test scripts.
- FI-Contributor needs to be able to add, edit, modify and delete interoperability test scripts. Several programming languages should be supported.
- FI-Contributor may have means of providing other necessary software for the test scripts execution.

3 Requirements for performance and privacy testing

Performance testing concerns the behavior of a system under a given work load and/or network conditions. It includes scalability, resiliency, energy efficiency and QoS/QoE as discussed in the following sections.

Privacy testing aims to evaluate the security vulnerability and the privacy leakage at any component (IUT) of the system, and therefore, the security/privacy risks of the whole SUT, as described in detail in Section 3.3. *Note: the requirements described here relate only to the IUTs and SUT of individual tests; privacy and security of the F-Interop-Platform itself will be described in a separate document (F-Interop deliverable D1.2).*

3.1 About scalability testing

3.1.1 Main architectures and approaches for scalability testing

The purpose of scalability testing is to qualify and quantify the effect of a certain workload on an IUT. A scalability test defines the aspect on which load is applied, KPIs and a PASS/FAIL criteria. A performance test (such as a scalability test) without predefined PASS/FAIL criteria is called a **benchmark**. The results from a series of benchmarks tests can be plotted and analyzed to reveal the critical points, where the behavior of the system changes significantly, which are usually also infliction points in the resulting graphical representation.

Scalability testing is usually application specific. Generating transactions, for example, requires software, which is written to emulate real users or IoT devices. Scalability testing commonly involves active emulators that provide stimulus to the system and also measure the response. Scalability testing might also be performed in a passive monitoring mode - multiple IUTs are instantiated and the metrics are measured by applying passive traffic analysis. Scalability testing can be performed in two ways:

- **Single implementation test**: the test bed consists of an IUT and one or more emulators.
- **System performance test:** the test bed consists of one or more emulators and an SUT (system under test), which may be a complete or a partial network.

Scalability tests are best performed with a single degree of freedom (one free variable), while all other configurable parameters are bound variables. There is also a distinction between attempted or targeted value and the observed value. In many cases, the target can directly influence the results - when attempting a rate which is much higher than the capacity of the system, it may perform worse than the maximum the system is capable of.

Common scalability variables are:

- Target throughput
- Attempted transaction rate

- Target number of concurrent connections
- Attempted rate of connection setup/tear-down

Common measurements for scalability tests are:

- Achieved throughput
- Achieved transaction rate
- Maximum number of concurrent connections
- Achieved rate of connection setup/tear-down
- Connection latency
- Transaction latency

When a threshold is specified to a measured value, a performance test can be conducted in an explorative fashion by increasing or decreasing the free variable and recording the maximum (or minimum) value while the test conformed to the specified threshold. A common approach is to use the binary search algorithm on the free variable.

For example, a scalability test can be designed to determine the effect of a high number of connections on an IUT. The number of connections is where the load is applied, while the effect is measured by connection latency, transactional latency and overall throughput.
The test can be conducted in three ways:

- **Verification:** single test run with a predetermined target values and expectations.
- **Benchmark:** multiple test runs are performed with a different target value for the number of connections attempted. The results are graphed by plotting the attempted number of connections against the measurement data observed for each test run. A human interpreter on the graphical representation of the results performs analysis.
- **Binary search:** a threshold is defined for some or all of the measured values (e.g., maximum connection latency, maximum transactional latency) along with the maximum number of iterations. The binary search algorithm governs the target number of connections attempted: when the test results exceed the thresholds, the target number is reduced; when the test results conform to the thresholds, the target number is increased. This is performed until the number of test runs exceeds the maximum number of iterations.

3.1.2 The three main steps in scalability testing

The **first step** in scalability testing is to specify the test goal, test setup, performance parameters and performance expectations. For example:

> **Goal:**
> Measure the transactional latency of the IUT under high connection load. Use binary search to determine the maximum connection load while conforming to the performance expectations.
>
> **Test setup:**
> The IUT is a CoAP server.

CoAP clients are emulated.
The test setup also includes a description of the topology.

Performance parameters:
Concurrent connections: [10 to 10,000] connections
Connection setup rate: 10 new connections per second
Transaction rate per connection: 1 transaction/sec
Multiple performance parameters are possible; One or zero parameters should describe a range, while all others have constant values.

Performance expectations:
Transactional latency < 100ms
Maximum test run duration: 2000 seconds
*Multiple performance expectations are described. A result considered to pass the test must conform to **all** performance expectations listed.*

The **second step** is to instantiate the test environment, including the configuration of IUT/SUT and emulators, i.e., to implement the **test setup** part of the test case description. The emulators are then configured according to the performance parameters and performance expectations.

The **third step** is to execute the test once (when performing a verification test) or multiple times (when using a variable) and record the measurements for each test run. The measurements are then used either to directly evaluate the test run (pass/fail) or to provide a graphical representation of the performance response of the system.

3.1.3 State of the art on scalability testing

Scalability is recognized as a deciding factor in many applications and is tested throughout the development lifecycle of an implementation:

- Early development phase - Many vendors integrate full scalability tests and unit tests as part of a continuous development process. Such a vendor defines salient metrics and work loads according to the specific product and use cases, allowing it to test the effect of code changes on the performance of the implementation.
- Release engineering and marketing - Results of scalability tests are used to create the end user specifications for new releases on an implementation.
- Customer testing (PoC, ATP) - Customer oriented testing determine the applicability and scalability gap of the implementation before deploying into service or for regression testing of new software releases.

High-end test equipment is usually needed to generate a high load while emulating real network conditions, since the test equipment should over-perform the implementation it is testing.
In the case of a complete solution under test, an appropriate test bed setup must be designed and implemented, which includes configuration of the implementation under test and the test equipment.

The execution of these tests takes place in a lab (vendor's lab, customer's lab or an independent lab) and representatives from all the involved parties are commonly present on-site during the critical phases of the testing.

Scalability testing is generally cost and effort intensive: other than the testing phase itself, the costs occur due to logistics, leasing of equipment if needed, travel, time scheduling and the design and configuration of the test bed.
The high cost associated with this form of testing may prove prohibitive to SMEs; the TBaaS approach in F-Interop should aim to alleviate this and make scalability testing available and affordable to SMEs.

3.1.4 Towards key requirements identification for remote online scalability testing

Following the discussion in the introduction (see Section 1.2.3.2), we have chosen to focus on the application layer protocol CoAP for remote scalability testing.
As a secondary goal, we also intend to design the tools to be modular by separating the generic part from the protocol specific part in order to maximize the reusability for other protocols and reduce the effort for extending the platform to new standards.

3.1.5 FI-User and FI-Contributor needs for remote online CoAP scalability testing

This section synthesizes the general approaches for scalability testing to remote online CoAP scalability testing. Specifically, it addresses the following scenarios:

- One-to-many scalability tests with passive monitoring, when the user initiates the interaction to the test bed.
- One-to-many scalability tests with passive monitoring, when the test bed devices initiate the interaction to the user's IUT.
- Many-to-many scalability tests with passive monitoring.
- One-to-many scalability tests with active emulation.

3.1.5.1 FI-User and FI-Contributor needs for scalability tests with passive monitoring

One-to-many tests can be conducted in two forms. Either the user controlling the IUT initiates the interaction to the many devices in the test bed or the user requests the many devices in the test bed to initiate the interaction to the user's IUT. Both forms employ passive monitoring to analyse the interaction, regardless of the identity of the initiating party.
The implementation may be hosted by the user and connected via tunnels or deployed into the F-Interop-Platform.

Many-to-many tests are conducted between two groups of devices. At least one group of devices is hosted in an F-Interop test bed, while the other group may be hosted locally be the user or deployed on an F-Interop test bed.

FI-User and FI-Contributor needs for scalability tests with passive monitoring:

- FI-User should have a simple method of remotely connecting his/her IUT to the FI-Platform.
- FI-User should have a simple method of deploying his/her implementation in the FI-Platform.
- FI-User must be able to interact with his/her implementation when he/she deploys it in one of the testbeds of the FI-Platform.
- FI-User should be provided with a real-time traffic analysis tool.
- FI-User needs to be able to set performance parameters and expectations for each test.

- FI-User must be able to select which performance data is collected.
- FI-User must be able to launch and abort a test.
- FI-User must be provided with results of the tests after the tests are completed. The results must include the parameters of the test that allow a complete reproduction of the test.
- FI-Contributor should be able to provide new passive monitoring applications for additional analysis functionality of CoAP or analysis of other protocols

3.1.5.2 FI-User and FI-Contributor needs for scalability tests with active emulation

The active emulation tests will leverage the passive monitoring facilities and add the emulation functionality. The FI-User and FI-Contributor needs are the same as for the *FI-User and FI-Contributor needs for scalability tests with passive monitoring* described above with the addition of the following needs:

- FI-User must be able to control scalability parameters of the emulators (e.g., number of emulated clients, per client transaction rate).
- FI-User needs to be able to control the timeline of the test before launching the test.
- FI-Contributor should be able to provide new active emulation application for additional functionality emulation of CoAP or emulation of other protocols.

3.2 About resiliency testing

3.2.1 Main architectures and approaches for resiliency testing

Resilience refers to the behavior of an implementation under sub-optimal conditions. The common scenarios for such conditions include:

- Link failure and recovery
- Node failure and recovery
- Fail-over
- Continuous and intermittent packet loss
- Packet duplication
- Delay and delay variation
- Recovery cascade (applicable for large scale deployments)

When measuring network infrastructure (i.e., routers, switches, firewalls, etc.), the applicable scenarios are link failure, node failure and fail-over scenarios. For applications, the applicable scenarios are those that emulate such infrastructure failures: packet loss, delay and recovery. The approach to testing varies according to different cases.

3.2.1.1 The case of infrastructure failures

Infrastructure failures include link and node failures as well as fail-over. A fail-over is a special case of a link/node failure, where a network element (such as a firewall) is protected by a peer network element.
For failure conditions on the infrastructure level, the outcome of the test is expressed in out of service time. The definition of out of service time may vary according to the scenario. In the simplest variant, out of service time is measured on the data plane:

the tester generates a data stream, an impairment device then emulates a failure; the time until full recovery is measured by calculating the packet loss according to the transmission rate.
When testing applications, the out of service time can be defined in various ways. For example, an application emulator generates user traffic followed by an emulated failure. The timestamp of the emulated failure is compared with the timestamp of the first application layer response (which must commonly be a valid response and not an error report).

The key indicator for infrastructure failure mitigation is the out of service time expressed in seconds.

3.2.1.2 The case of packet loss resiliency

The tolerance of packet loss varies from application to application. For real time applications, such as VoIP and video conferencing, packet loss cannot be easily mitigated, since the data carried by the packets is valid for a short period of time. Other applications provide reliability through retransmission of messages. CoAP provides reliability through a mechanism called Confirmable Messages.

Emulation of varying degree of packet loss allows studying the tolerance thresholds of an application. For example, a reliable application might have a considerable overhead under high packet loss caused by the constant retransmissions. For an unreliable application, the deciding factor is the degradation level of the user experience.

The key indicators for packet loss tolerance for applications with reliability mechanisms are:

- Traffic overhead: how much additional traffic is transmitted under a certain packet loss pattern and a certain packet loss rate.
- Amount of lost data: how much data is lost due to re-transmission failure

The key indicators for packet loss tolerance for applications with reliability mechanisms are:

- Quality of experience markings (e.g., MOS for VoIP)

3.2.1.3 The case of packet duplication resiliency

The impact of packet duplication is application dependent. Generally, TCP based applications are not affected by packet duplication. On the other hand, UDP based applications may be impacted. To bridge the gap, UDP based applications may implement a message ID mechanism. For example, CoAP, which is a UDP based protocol, specifically describes a message de-duplication process.

The key indicators for packet duplication resiliency are application dependent and may rely on QoE metrics (e.g., MOS for VoIP). Specifically for CoAP, the requirement is a conforming behavior to the message de-duplication procedure described in the standard.

3.2.1.4 The case of delay and delay variation tolerance

The impact of delay and delay variation is also application dependent. All applications that use TCP are impacted by delay and delay variation - due to the window acknowledgement mechanism, high delay corresponds with lower throughput and delay variation corresponds with throughput instability.
Other applications are time sensitive and have a hard limit on the amount of delay and delay variation they may tolerate.

The key indicators for packet delay and delay variation resiliency are application dependent and may rely on QoE metrics (e.g., MOS for VoIP). When an application is time sensitive and poses hard limit on the amount of delay and delay variation, the key indicator is correct behavior under the same limits. For example, if an application specifies a tolerance up to 250 ms and ± 70 ms delay variation, it should be tested under these conditions, i.e., by introducing a constant delay of 250 ms and a variation of ± 70 ms.

3.2.1.5 The case of recovery cascade

When a high number of devices communicate with a system and failure occurs, a recovery cascade occurs following the recovery from that network failure. In contrast to scalability testing, where the number of clients increases gradually, in a recovery cascade all clients are already instantiated and may also have a state such as an open connection or an ongoing transaction.

The key indicators for a successful mitigation of recovery cascade are:

- No fatal error conditions due to sudden overload (e.g., software crash, database deadlock)
- Low effect to transactional latency
- Most clients resume normal operation seamlessly. Clients that do not resume normal operation seamlessly, resume normal operation after retrying to re-establish communication with the system.

3.2.2 The three main steps in resiliency testing

The **first step** in resiliency testing is to specify the test setup, the type of emulated network condition and the key performance indicators.

The **second step** is to setup the network scenario and model the network conditions using a specialized impairment device or software.

The **third step** is to emulate the desired network conditions and collect the key performance indicator values.

3.2.3 State of the art on resiliency testing

The emulation of the various network conditions described in the preceding sections is performed by means of a specialized hardware called a network impairment device. Network impairment can also be performed in software when the amount of

delay variation is not stringent. Some network conditions are emulated manually: link failures are emulated by physical disconnection and re-connection while removing and restoring power supply emulate node failures.
Similar to scalability testing, resilience testing, execution of these tests takes place in a lab (vendor's lab, customer's lab or an independent lab) and representatives from all the involved parties are commonly present on-site during the critical phases of the testing.

3.2.4 Towards key requirements identification for remote online resiliency testing

In the scope of F-Interop, we have chosen to focus on relevant scenarios applicable to IoT and implement the emulation of network conditions in software. The software will provide packet loss impairment, insertion of delay and delay variation as well as packet duplication.
Recovery cascade can be emulated by combining packet loss emulation and a scalability tester.
We consider infrastructure failures to be out of scope for F-Interop.

3.2.5 FI-User and FI-Contributor needs for remote online resiliency testing

All of the chosen scenarios for remote online resiliency testing can be covered by a single test approach, namely the usage of an impairment tool implemented in software.

The impairment tool can only apply the selected model for traffic flowing through it. In other words, the network must be configured to divert traffic into it and it must act as a transparent bridge.

FI-User and FI-Contributor needs for remote online resiliency tests:

- FI-User must be able to set one or more models of impairment via FI-Platform.
- FI-User must be provided with statistics regarding the impairment, i.e., the number of packets and bytes that were matched for each impairment profile (drop/reordering/latency).
- FI-User should be provided with real-time statistics regarding the impairment.
- FI-User must be able to enable and disable the impairment after setting up the test scenario.
- FI-Contributor should be able to provide additional impairment modules which implement extended functionality (such as application-specific impairment).

3.3 About security and privacy risk testing

3.3.1 Main approaches for security and privacy risk testing

The purpose of security/privacy risk testing is to evaluate the security vulnerability and the privacy leakage at an IUT, and thus, of the whole SUT. In fact, the wide adoption of IoT sensors and pervasive networked computing devices that process privacy sensitive data leads to the fact that a single security vulnerability in a single

component may compromise the whole system. Similarly, a privacy leakage at any of the components may affect privacy protection of the whole system.

The term *privacy* refers to the ability to choose what data to expose to others and what data to keep from others. Thus, privacy gives to users control over their data disclosure. When interactions happen via computers and networks, privacy relies on technical tools for data confidentiality (i.e., rules that limits access to information), and data integrity (i.e., trustworthy and accurate information).

On the other side, an attacker, by using different tools and mechanisms, can obtain data, which should be kept confidential. For instance, by using traffic analysis, a local eavesdropper may attempt to infer information about the content of encrypted and/or anonymized connections by observing traffic patterns. The attacker can use metadata such as packet size and the direction of the traffic flow without breaking the encryption.

Besides traffic analysis, another approach for testing the security vulnerability of a system consists in checking user profile policies and verifying, for instance, if users are given access to resources/data when it should be denied.

Moreover, anomalies in the networks such as abnormal energy consumption on an IoT sensor can be considered as symptom of node attacks and thus, security risks.

3.3.2 The three main steps in security and privacy risk testing

The **first step** in security/privacy risk testing consists in setting up the test scenario (IUTs, SUT and the type of traffic to collect for further analysis).

The **second step** consists in collecting the data. The dimension of the data set is quite important, since it impacts the probability that an attacker can extract useful information by looking for traffic patterns.

The **third step** consists in data extraction and processing. The attacker may be interested in identifying the end user, finding out about his/her location, the services he/she is usingand the confidential information he/she is exchanging.

3.3.3 State of the art on security and privacy testing

Due to the insecure wireless channels and constrained resources, IoT devices are extremely vulnerable to various types of threats and attacks. Privacy preservation becomes extremely important especially when IoT networks are deployed to monitor valuable asset or military targets. An adversary with the knowledge of the location of the data source or base station location may be able infer the content of the data being transmitted or destroy the monitoring objects.

In the literature, privacy threats have been classified into content privacy and contextual privacy. For the content privacy threat, the adversaries try to capture the packets in the networks to obtain sensitive data. Encryption technologies are used to counter this privacy threat. For the contextual privacy threat, the adversaries attempt

to locate the position of some vital targets by eavesdropping on the networks and by using traffic analysis techniques.

3.3.4 Toward Key requirements identification for online security and privacy testing

In the context of F-Interop we will mainly focus on traffic analysis techniques, aiming to extract not only user/node location, but also other personal information. Moreover, we intend to explore security/privacy risks, by checking user profile policies (with QoS SDN/NFV-based tests) and anomalies in the networks (for instance with resiliency testing – node/link failure may be due to an attack, and with energy efficiency testing – abnormal energy consumption, as symptom of a node attack).

3.3.5 FI-User and FI-Contributor needs for online security and privacy testing

- FI-User should have a simple method for remotely connecting his/her implementation to the FI-Platform.
- FI-User must have access to the data exchanged on the encrypted channel between the IUT and the FI-Platform.
- FI-User should be able to select among different traffic analysis tools/methods.
- FI-User should have access to the data set used for traffic analysis.
- FI-User should be provided with the results of the test, once the traffic analysis has been completed.
- FI-User may be provided with a privacy-visualization tool for showing tests results verdicts in “real-time” (PASS, FAIL)
- FI-User must be able to launch and abort a test.

3.4 About energy efficiency testing

3.4.1 Main architectures and approaches for energy efficiency testing

The most accurate way to measure energy efficiency is a direct measurement with a physical meter device or by using the data reported by the device's power supply unit. However, by leveraging general knowledge of computational systems, one can define a model that delivers a close estimation of the energy efficiency without requiring a physical meter device. Such model is especially suite for remote online testing. There are a number of possible models for energy efficiency estimation:

- **Device-independent (black-box) estimation:** performed by taking the data transfer statistics (input/output bits) and user specified µW/bit (micro-Watt per bit). The µW/bit parameter may be benchmarked in advance by the user or may be specified by the chipset vendor.
- **Emulator data:** some IoT implementations run on operating systems that also allow software emulation of the device. One such operating system is Contiki, an operating system that also delivers system power consumption estimation.

Power consumption measurement or estimation of a device in idle mode (or standby) does not provide enough information to produce a complete picture of the energy

efficiency, since the power consumption depends on the workload. Therefore, energy efficiency testing must be combined with other forms of testing such as scalability and resiliency.

3.4.2 The three main steps in energy efficiency testing

The **first step** is to define a workload for the device using a scalability or resiliency scenario as well as define the power consumption estimation model.

The **second step** is to setup the scenario and emulate the workload while performing the measurement according to the power consumption estimation model.

The **third step** is to correlate the measurement with the workload and model. The output is the estimated power consumption.

3.4.3 State of the art on energy efficiency testing

As discussed above, energy efficiency testing is usually performed with physical instruments or direct data from the power supply units of the device. The execution of such tests takes place in a lab (vendor's lab, customer's lab or an independent lab).

3.4.4 Towards key requirements identification for remote online energy efficiency testing

In the scope of F-Interop, we have chosen to focus on modeling the energy efficiency, since it poses no requirements for physical availability and provides a general solution.
For a device independent (black-box) power efficiency estimation, we will use traffic statistics combined with a power consumption estimation expressed in µW/bit.
We also intend to explore the possibility of gathering information directly from IoT operating systems such as Contiki OS and OpenWSN.

3.4.5 FI-User and FI-Contributor needs for online energy efficiency testing

To allow alternative models to be applied to energy efficiency testing, it is beneficial to design the energy efficiency estimation as modular plugins. The F-Interop-Platform must provide a plugin for the general case, which will be based on traffic analysis. Each plugin must provide a metadata specification. For example, the traffic analysis plugin will have a "µW/bit" field. Another plugin might be based on processor sleep states and will have a "mW/sec" field for each sleep state.

FI-User and FI-Contributor needs for remote online energy efficiency tests are:

- FI-User needs to be able to provide power consumption estimation based on metadata of an energy consumption plugin.
- FI-Contributor should be able to integrate implementation specific power consumption estimation plugin into the FI-Platform.

3.5 About QoS SDN/NFV-based testing

3.5.1 Main architectures and approaches for QoS SDN/NFV-based testing

The purpose of QoS SDN/NFV-based testing is to qualify and quantify QoS performances of a given system, by using the SDN paradigm for collecting statistics related to node and links, and NFV approach for emulating network conditions.

Software Defined Networking (SDN) separates the data plane (i.e., traffic forwarding between network devices, such as switches, routers, hosts, etc.) from the control plane (i.e., the decision making about the routing of traffic flows), and allows programmability of the network by external applications. By means of a Southbound Interface (SBI) and a Northbound Interface (NBI), the control plane can interact, respectively, with the data plane and the application plane. Since SDN's birth, the SBI has been standardized and commonly identified in the OpenFlow protocol, while the NBI is still under development.

Thus, the main components of SDN architecture are the SDN controller and the OpenFlow switches.

Each OpenFlow switch contains one or more *Flow Tables*, which perform packet look-ups and forwarding. Each table contains *Flow Entries*, which are used as rules to process matching packets. When an incoming packet is not registered in their flow table, the switches send a query to the controller. The controller can add, update or delete flow entries in the flow table enabling it to control flow traffic in the network. Each flow entry mainly consists of match fields that define the flow, instructions describing what to do with the matched packets and counters for statistical purpose.

The controller can impose policies on the network traffic flows – such as sending data by routes that are faster, less congested or have the fewer number of hops – in order to provide different QoS levels to different classes of traffics. OpenFlow can control traffic on the basis of various routing granularities such as destination IP address, TCP flows, etc.

Several open-source SDN controllers have been developed, such as OpenDayLight, Flood-Light, and POX. OpenDaylight is a community-led, open, industry-supported framework for accelerating SDN adoption. The OpenDaylight controller is part of the OpenDaylight framework. The controller provides base network service functions like network topology and network statistics. The latter can be used for evaluating QoS performance.

Network Function Virtualisation (NFV) is an architecture that enables the dynamic management of network services life cycle while relying on virtualised infrastructure. NFV is currently in a process of standardization by ETSI NFV ISG, whose focus is on the management and orchestration aspects. The current state of the standardization covers a high-level specification of interfaces between the management and orchestration functional blocks. It specifies how network services, composed of virtual network functions (VNFs), are orchestrated and managed. Since the ETSI NFV ISG specifications do not currently address the protocol level of interaction between the blocks, in the framework of F-Interop we shall apply the NFV approach of using VNFs

to provide testing modules, while the F-Interop orchestration layer performs the management and orchestration.

Since we cover emulation of network conditions in resiliency testing, this section focuses on collection of statistics by using passive and active methods.
The **passive** method relies on the collection of QoS statistics from SDN switches by setting up a classification counter of the flow between two implementations.
The **active** method relies on the usage of agents (deployed as VNFs) co-located with each implementation (under test or otherwise), allowing measuring QoS parameters between each endpoint. The data provide by the probes is especially useful for remote online tests, where the implementations may be distributed in various geographical locations (one or more remote testing labs and one or more user locations). In addition to the probes, the SDN controller enables discovering the topology through one or more SDN switches to identify paths with sub-optimal QoS parameters.

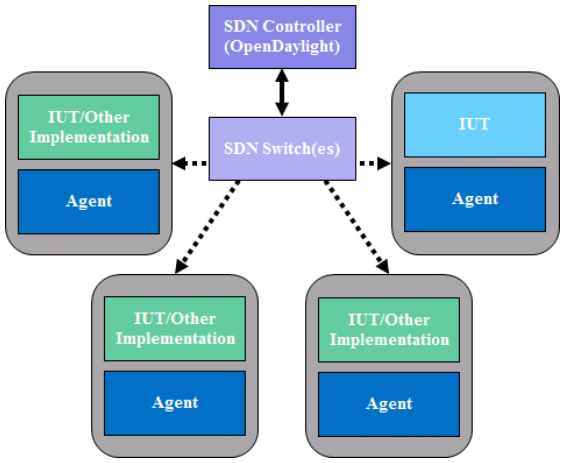


Figure 7 - QoS SDN/NFV-based Testing Topology

The following statistics are collected in the passive method are:

- Throughput
- Packet loss

The statistics collected by the active method are:

- Synthetic packet loss
- Synthetic latency
- Synthetic latency variation

Note that the measurements performed by the probes are all synthetic, i.e., they are performed by generating additional traffic, rather than analysing user traffic.

3.5.2 The three main steps in QoS SDN/NFV-based testing

The **first step** in QoS SDN/NFV-based testing consists in specifying the test setup, the network scenario (i.e., the SDN controller and the location of the Agents), and the QoS metrics in scope of the test.

The **second step** consists in setting up the network scenario and configuring the agents.

The **third step** consists in collecting the passive measurements from the SDN controller and/or the active measurements from the Agents, and evaluating the SUT's performance. The results can be plotted and visualized.

3.5.3 State of the art on QoS SDN/NFV-based testing

In order to evaluate QoS performance, several monitoring and measurement techniques have been designed for computer networks. They can be classified into two groups: passive and active. Passive methods simply listen to network traffic and collect related information. Instead, active methods inject new traffic (i.e., probe packets) into the network, and performance metrics are derived, based on the injected data (e.g., delay encountered by the probe packets).

Many tools [7] dealing with network monitoring and analysis have been developed. Statistics from probing tools like ping, traceroute, tcpdump, and nmap were traditionally used for performing network monitoring and detecting network issues. Nowadays however they are becoming insufficient for large and complex infrastructures supporting the exchange of high amounts of traffic.

A similar approach to active measurements is taken by a distributed monitoring platform, called v6Sonar [8], including several monitoring agents and a controller. v6Sonar Agents can collect different data useful for calculating synthetic user experience for specific services. Such data can convey valuable information on how users experience an online service. Degradation in the perceived service implies potential issues in the network infrastructure, which is then not able to offer the due QoS.

The Software Defined Networking (SDN) paradigm is emerging as a promising solution for simplifying network management, which is traditionally based on manual intervention of network administrators for future large heterogeneous networks. SDN allows highly dynamic, flexible and automated configuration of the network, more efficient use of network resources, and easier way for operating, troubleshooting and debugging. SDN and in particular the OpenFlow protocol allow for better oversight capabilities than what existed in traditional networks. Such capabilities can be used for real time monitoring and reaction to network events. For instance, global statistics and dynamically adjustable granularity of rules installed in response to a network event come easier in SDN networks. Unlike before, when administrators had to manually log in into each network device and understand its state, with SDN features such statistics can be delivered centrally and correlated with other information relevant to operations.

OpenNetMon [9] is a monitoring tool for OpenFlow networks, recently proposed in literature, which is able to monitor per-flow end-to-end (e2e) QoS parameters (i.e., throughput, delay and packet loss) by pulling the edge switches along the path at an adaptive rate. Such active measurements can be used as input for Traffic Engineering (TE) techniques, to compute end-to-end paths, and properly distribute traffic load into the network to achieve the aimed QoS.

SLAM, Software-defined Latency Monitoring [10], is a path latency monitoring framework for SDN data centers. SLAM uses timestamps of carefully triggered control messages to monitor network latency between any two arbitrary switches and identify high-delay paths. Traffic can then be routed far away from such high-delay path segments.

FlowSense [11] is a monitoring tool that computes network link utilization (i.e., throughput) with zero overhead, using the control messages exchanged between the OpenFlow switches and the controller. Despite the advantage of introducing zero cost (and zero overhead), since it is fully based on passive measurements, the statistics information may be inaccurate when the controller does not often receive OpenFlow messages from the switches. Another low-cost but high-accuracy flow monitoring scheme is FlowCover [12]. It reduces the monitoring cost by aggregating the polling requests and replies and by selecting the target switches based on active flows rather than on a generic per-flow basis.

SDN-RADAR [13] is a troubleshooting tool that combines the v6Sonar Platform, with the SDN architecture. By leveraging on the end-to-end latency measurement provided by the v6Sonar Agents, and the topology information available at the SDN controller, SDN-RADAR can monitor QoS, compare it to the SLA and localize underperforming links (i.e., links which are experiencing latency and/or packet loss higher than the admitted threshold value).

3.5.4 Toward key requirements identification for remote online QoS SDN/NFV-based testing solutions

F-Interop will focus on Quality of Service (QoS) tests based on SDN/NFV architecture. In other words, F-Interop will develop QoS tests that exploits the advantages offered by SDN (in term of monitoring QoS metrics), and by NFV for deploying agents and for emulating network conditions. (Note that emulation of network conditions is covered under resiliency testing, section 3.2).

3.5.5 FI-User and FI-Contributor needs for remote online QoS SDN/NFV-based testing

Both active and passive methods of QoS SDN-NFV based tests apply to the different scenarios we considered for F-Interop and are designed for many-to-many tests. One-to-many and one-to-one tests are considered as specific cases of many-to-many, and do not require a different approach, components or needs.

FI-User and FI-Contributor needs for QoS SDN/NFV based tests are:

- FI-User must be able to connect his/her IUT(s) to the FI-Platform
- FI-User must be able to access a SDN controller if available in the F-Interop Platform
- FI-User must be able to deploy a SDN controller module within the F-Interop Platform if not available yet

- FI-User should be able to select which scenario he/she wants to use in the performance test session.
- FI-User must have a simple way for connecting the VM to the SDN controller.
- FI-User should be able to deploy Agents on the IUT and/or other devices, for injecting probes into the network
- FI-User must be able to retrieve the statistics collected by the SDN controller.
- FI-User must be able to retrieve the latency measurements collected by the Agents
- FI-User should be able to select which kind of QoS metrics (latency, packet loss, throughput) he/she wants to tests
- FI-User should have an interface displaying the collected statistics

4 General needs for online interoperability and performance testing

Based on the lessons learnt from all the considered uses cases in previous sections, we came to identifying the set of actions that an F-Interop-User has to perform in order to execute remotely interoperability and performance tests using the F-Interop-Platform. This set of actions is called an "**F-Interop session**" and it is summarized in the table below.

Step	Action	Description
0	FI-User authentication and authorization. IUT registration / identification	FI-User authenticates in a secure way (prior FI-User registration needed) in FI-Platform. FI-User needs to be authorized to use FI-Platform resources. FI-User identifies which IUT he/she will test (prior IUT registration needed).
1	Test suites discovery and selection	FI-User starts by discovering the available test suites and by selecting the one he/she wants to execute.
2	Resource description	FI-User specifies/selects resources in the F-Interop-Platform that are needed for his/her F-Interop session including the location models[1], testing tools, libraries, etc. During this phase FI-Platform may request information from FI-User or provide information to FI-User for a coherent selection of the required resources.
3	Resource reservation	The resources selected in the previous step are actually reserved.
4	Resource provisioning, configuration and session setup	The instantiation of the F-Interop-Platform resources that fit best with the FI-User needs is done.
5	Test execution	The online F-Interop test campaign is launched and the selected (executable) test suites are executed against the IUTs.
6	Results analysis and report	Test execution information is analyzed. The test results and verdicts are provided together with explanations in case of FAIL or INCONCLUSIVE verdicts or something wrong happened. A report can be provided under request in case for example the FI-User wants

[1] Location models are the different configurations for the location of components of the test. These will be defined in D1.3.1. [6]

		to apply for a certification/labelling program.
7	Session storage	Storage of the F-Interop session information (Session-id, User-id, FI-User's IUT-id, IUTs' version, test description, test version, testing tool, test log and results, etc.). This has to remain accessible beyond the F-Interop session for the involved parties.

Table 1 - F-Interop Session

Using this description of F-Interop session combined with the previously gathered needs we synthesized and organized FI-User and FI-Contributor needs for online interoperability, conformance and performance testing in Table 3 and Table 4.

5 Requirement Synthesis and Analysis

The purpose of this report “D1.1 – Requirements specification report” is to describe requirements from the F-Interop-Platform point of view that enable and ease online interoperability and performance testing. In this section we first summarized the needs gathered in previous sections in shape of tables. During the analysis of those needs, some appeared in all the considered use cases. They have been regrouped in the tables: F-Interop-User needs (see Table 3), and F-Interop-Contributor needs (see Table 4). We use the following naming rule to identify the needs: ***FI-UN.XX*** corresponds to the XXth need from the F-Interop-User point of view, while ***FI-CN.XX*** represents the XXth need from the F-Interop-Contributor point of view.

From these needs, we synthesized the requirements for the F-Interop-Platform and presented them in Table 5 using the following naming rule to identify the requirements: ***FI-PR.XX*** is the XXth requirement for the F-Interop-Platform.

For these F-Interop-User needs, F-Interop-Contributor need and F-Interop-Platform requirements, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC-2119 (IETF). The key words used in the specification to indicate requirement levels are summarized in the following table:

MUST	This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase “SHALL NOT”, means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label
MAY	This word, or the adjective “OPTIONAL”, means that an item is truly optional.

Table 2 - Key words indicating requirements levels

5.1 Collected F-Interop-User needs

The following table summarizes the gathered F-Interop-User requirements.

F-Interop-User needs			
Id	Field	Scope	Description
FI-UN.01	FI-Session.0	FI-User authentication and authorization	FI-User MUST be authenticated in a secure way in the FI-Platform and authorized to use the available resources.
FI-UN.02	FI-Session.0	IUT registration / identification	FI-User MUST register the IUT(s) prior to executing a FI-Session. FI-User MAY select a previously registered IUT.
FI-UN.03	FI-Session.1	Access to test descriptions	FI-User MUST have access (view) to the available test suites descriptions.
FI-UN.04	FI-Session.1	Access to different versions of test descriptions	FI-User MUST have access (view) to the older versions of test suites descriptions.
FI-UN.05	FI-Session.1	Test suites discovery and selection	FI-User MUST be able to select the test suite to be executed in the FI-Session.
FI-UN.06	FI-Session.1	Test suites discovery and selection	All during the first phases of F-Interop Session definition, FI-User MUST be provided with thorough information such as the methodology for running the F-Interop session, the available location models, the resources needed for executing the test (software, sniffers, etc.), the type of test suites available and the automated IUTs available in F-Interop-Platform.
FI-UN.07	FI-Session.1	Resource description	For the case of performance testing, FI-User SHOULD be able to select which kind of QoS metrics (latency, packet loss, throughput) he/she wants to tests
FI-UN.08	FI-Session.1	Resource description	For the case of performance testing, FI-User MUST be able to set performance parameters and expectations for each test.
FI-UN.09	FI-Session.1	Resource description	For the case of performance testing, FI-User MUST be able to select which performance data is collected.

FI-UN.10	FI-Session.1	Resource description	For the case of performance testing, FI-User MUST be able to control scalability parameters of the emulators (e.g., number of emulated clients, per client transaction rate).
FI-UN.11	FI-Session.1	Resource description	For the case of performance testing, FI-User MUST be able to control the timeline of active performance tests using emulators before launching the test.
FI-UN.12	FI-Session.1	Resource description	For the case of performance testing, FI-User MUST be able to set one or more models of impairment via FI-Platform.
FI-UN.13	FI-Session.2	Resource description	For the case of performance testing, FI-User MUST be able to provide power consumption estimation based on metadata of an energy consumption plugin.
FI-UN.14	FI-Session.2	Resource description	FI-User SHOULD have an easy-to-use interface for configuring the F-Interop session. Via this interface FI-User MUST be able to select which test cases he/she wants to execute. During this phase FI-Platform may request information (including the location models, testing tools, libraries, etc.) from FI-User or provide information to FI-User for a coherent selection of the required resources.
FI-UN.15	FI-Session.3	Resource reservation	FI-User MUST be able to access the SDN controller if available in F-Interop platform. If not available, FI-User MUST be able to deploy and configure an SDN controller in F-Interop platform.
FI-UN.16	FI-Session.3	Resource reservation	FI-User MUST be provided with means to connect VM(s) to the SDN Controller of F-Interop Platform.
FI-UN-17	FI-Session.3	Resource reservation	FI-User SHOULD be able for injecting probes into the network like Agents on the IUT(s) and/or other devices.
FI-UN.18	FI-Session.3 and FI-Session.4	Resource reservation, resource provisioning and session setup	After validating resource description, FI-User MAY be provided with information about the automatic resource reservation, provisioning and session setup. Information such as estimated time for completion or problems/error encountered SHOULD be displayed.
FI-UN.19	FI-Session.4	Resource provisioning and session setup	FI-User MUST be able to deploy his/her IUT in the FI-Platform's testbeds, for certain location models. FI-User MUST be able to interact with the implementation once deployed.
FI-UN.20	FI-Session.5	Coordination of test	During the test execution, FI-User MUST have an interface displaying information

		execution	regarding the state of the automated test execution.
FI-UN.21	FI-Session.5	Coordination of test execution	FI-User MUST be provided with a messaging tool describing the actions needed from user's side (example "execute test TD_COAP_CORE_02").
FI-UN.22	FI-Session.5	Coordination of test execution	FI-User MUST be provided with means to control the execution using commands such as start, stop, skip test case, restart test suite.
FI-UN.23	FI-Session.5	Coordination of test execution	FI-User MUST be provided with means to abort an ongoing test execution.
FI-UN.24	FI-Session.5	Coordination of test execution	FI-User MUST have a text message based communication channel for coordinating execution of test with other FI-User when running interoperability sessions.
FI-UN.25	FI-Session.5	Test execution	FI-User MAY be provided with a visualizing tool for analyzing the traffic in real time.
FI-UN.26	FI-Session.5	Test execution	For interoperability testing, FI-User MAY be provided with a visualizing tool for showing test results verdicts while executing the test.
FI-UN.27	FI-Session.5	Test execution	For performance testing, FI-User MUST be able to enable and disable the impairment after setting up the test scenario.
FI-UN.28	FI-Session.6	Results analysis and report	FI-User MUST be provided with a test results report. Results verdicts should be provided along with explanations in case of FAIL or INCONCLUSIVE verdicts.
FI-UN.29	FI-Session.6	Results analysis and report	For interoperability testing, FI-User MUST be provided with means to apply for certification/labelling program.
FI-UN.30	FI-Session.6	Results analysis and report	FI-User SHOULD be provided with means to access the needed data exchanged during the test session to perform traffic analysis.
FI-UN.31	FI-Session.6	Results analysis and report	For performance testing, FI-User MUST be able to access/retrieve statistics collected by the SDN controller.
FI-UN.32	FI-Session.6	Results analysis and report	For performance testing, FI-User MUST be able to retrieve the latency measurements collected by the Agents.
FI-UN.33	FI-Session.6	Results analysis and report	For performance testing, FI-User MUST be provided with statistics regarding the impairment, i.e., the number of packets and bytes that were matched for each impairment profile (drop/reordering/latency).
FI-UN.34	FI-Session.7	Session storage	FI-User MUST be able to save F-Interop session information (session-id, description, IUT's is and version, test version, test log, results, etc).

FI-UN.35	FI-Session.7	Session storage	For interoperability testing, FI-User SHOULD be able to re-run test analysis from previously saved F-Interop Session traffic logs (without the need of performing the test again). FI-User SHOULD be able to select other tests cases implemented after the F-Interop session execution. This feature would be available only for certain interoperability passive testing tools.
FI-UN.36	FI-Session.7	Session storage	FI-User MUST be able view the information from saved sessions.
FI-UN.37	Reachability need / FI-Session.5	Remote execution of the F-Interop session	FI-User MUST be provided with tools that help it overcome typical reachability problems (like filtered UDP traffic on enterprise firewalls).

Table 3 - Collected F-Interop-User needs

5.2 Collected F-Interop-Contributor needs

F-Interop-Contributor needs			
Id	Field	Scope	Description
FI-CN.01	Test description	View / create / modify / delete test descriptions	FI-Contributor MUST have means to view / create / modify / delete the tests description.
FI-CN.02	Test description	View / create / modify / delete previously defined test descriptions	FI-Contributor MUST have means to view / create / modify / delete the tests description with versioning feature.
FI-CN.03	Test description	Provisioning templates for new test description	FI-Contributor MAY be provided with templates / models of test description when adding a new one.
FI-CN.04	Test tools	View, suggest modifications to test tools	FI-Contributor MAY be provided with means to view source code of testing tools and suggest modifications.
FI-CN.05	Test implementations	View / create / modify / delete test implementations	FI-Contributor MUST have means to view / create / modify / delete test implementations.
FI-CN.06	Automated IUT contribution	Upload an automated IUT contribution	FI-Contributor MUST have means to upload his/her own implementation under test (IUT) to the FI-Platform.
FI-CN.07	Test tools	Contribute with a testing tool	FI-Contributor MUST have means to implement and upload his/her own implementation of a testing tool.
FI-CN.08	Test implementations	View, suggest modification to test implementations	FI-Contributor MAY be provided with means to view code source of test implementations and suggest modifications.

Table 4 - Collected F-Interop-Contributor needs

5.3 F-Interop-Platform requirements

In the following table we present the derived F-Interop-Platform requirements (FI-PRs) from the previously gathered needs (FI-UN and FI-CN). This list of FI-PRs define which features and properties MUST/SHOULD/MAY be implemented in the F-Interop-Platform to satisfy the needs from F-Interop-User and F-Interop-Contributor and accomplish online interoperability, performances and privacy testing. The priority of the requirements is expressed keywords MUST/SHOULD/MAY as presented in Table 2.
As previously mentioned, each FI-PR is composed of five of elements: (1) Id, a unique identifier, (2) Field/Step, (3) the requirement scope - a statement describing a F-Interop-Platform function, a service provided or an operational constraint, (4) a detailed description of the requirement and (5) one or more references to FI-UN and/or FI-CN, providing the developer a traceable reference to a user's need that allows him to have verifiable/testable and user-oriented description of the functionality he is implementing.

F-Interop-Platform requirements				
Req. Id	Field/Step	Requirement Scope	Requirement description	Reference FI-UN/FI-CN
FI-PR.01	FI-Session.0	Authentication on the F-Interop Platform	FI-Platform MUST provide a unique and secure mechanism to register, authenticate and authorize an FI-User/FI-Contributor. FI-Platform MUST provide a mechanism to register an IUT. The IUT MUST be associated to the FI-User. During the registration FI-Platform SHOULD ask for information such as hardware used, protocol under test, protocol stack implemented, associated specifications of the standards implemented, IUT implementation version, etc. In FI-Session.0, FI-Platform MUST demand FI-User to identify the IUT to be tested.	FI-UN.01 FI-UN.02
FI-PR.02	FI-Session.1	Access to test description/ specification, with versioning features	FI-Platform MUST provide the list of available test descriptions for a given a protocol. As well as different versions of previous test description.	FI-UN.03, FI-UN.04, FI-CN.01
FI-PR.03	FI-Session.1	View, create, modify, delete features for test descriptions, with versioning features	FI-Platform MUST provide the means to view, create, modify and delete test descriptions, with versioning features. FI-Platform MUST provide templates and models when generating new test descriptions. FI-Platform must keep references to authors/contributors in these documents.	FI-CN.01, FI-CN.02, FI-CN.03

FI-PR.04	FI-Session.2 & FI-Session.3	Automated resource reservation, deployment and monitoring.	FI-Platform MUST automate resource reservation and deployment of both virtual and physical resources specified by the FI-User in FI-Session description. FI-Platform MAY provide information about the state of the operations (expected delays or errors/warnings found) while trying to deploy resources and also after they are deployed. For resource reservation and tools provisioning, FI-Platform SHOULD foresee problems and provide solutions for reachability/compatibility network problems. Example: remote user's IUT using IPv4 wont work with resources if deployed in testbed with public IPv6s.	FI-UN.18, FI-UN.19,
FI-PR.05	FI-Session.1	Test suites discovery and selection.	FI-Platform MAY provide networking compatibility and reachability tests adapted to the protocol testing solution and the location model chosen. FI-Platform MAY provide tools for overcoming most common compatibility and reachability problems –e.g. port forwarding via ssh tunneling for UDP-based test suites must be provided if FI-User is running his/her IUT behind a firewall that is filtering UDP traffic-	FI-UN.18, FI-UN.19
FI-PR.06	FI-Session.1	Test suit discovery and selection	FI-Platform SHOULD allow active and passive traffic analysis i.e. the use of Agents and probes installed in the IUT(s).	
FI-PR.07	FI-Session.1	Test suit discovery and selection	FI-Platform SHOULD support different tools to gather network traffic and statistics in real-time.	
FI-PR.08	FI-Session.1	Test suites discovery and selection	FI-Platform MUST provide the user with timeline control to allow the user to set scalability parameters for points in time relative to the beginning of test when using active protocol emulators for scalability tests.	FI-UN.22, FI-UN.23, FI-UN.24
FI-PR.09	FI-Session.1 & FI-Session.5	Test suites discovery and selection / Test execution	FI-Platform MUST provide the user with an impairment module that allows configuration multiple profiles to match traffic and to apply different impairment parameters for each profile. FI-Platform MUST allow applying packet loss to an impairment profile. FI-Platform MUST allow packet loss to be specified as a ratio from 100% to 0% (100% meaning complete packet loss and 0% meaning no packet loss).	FI-UN.25, FI-UN.33, FI-UN.27

			FI-Platform MUST allow applying latency and latency variation to an impairment profile specified in milliseconds. FI-Platform MUST allow applying packet duplication to an impairment profile specified as a ratio from 100% to 0% (100% meaning all packets are duplicated and 0% meaning no packets are duplicated).	
FI-PR.10	FI-Session.1	Test suites discovery and selection	FI-Platform MUST provide thorough information such as available tests suites, methodology for running the F-Interop session, available location models for each test suite, resources needed for executing the test (software, sniffers, etc.), automated IUTs available in F-Interop-Platform, during the first phases of F-Interop Session definition.	FI-UN.04,
FI-PR.11	FI-Session.1 & FI-Session.2	Test suites discovery and selection / Resource description	FI-Platform MUST provide a simple and interactive interface for configuring the F-Interop session. Via this interface FI-User MUST be able to select the resources needed for the F-Interop session, the test cases he/she wants to execute, the location models available for those tests suites, an automated IUT or other FI-User with whom he/she will be running the test in case of a IoP test, the testing tool, libraries, etc. FI-Platform interface SHOULD show options relevant to FI-User's previous choices, verifying they are coherent for running the FI-Session.	FI-UN.14 FI-UN.05
FI-PR.12	FI-Session.3	Resource reservation	FI-Platform MUST be able to host an SDN Controller. Its configuration and statistics MUST be accessible by any authorized FI-User/Fi-Contributor. FI-Platform MUST provides the right permissions to remotely access the SDN Controller in order to connect any VM(s) or other resources required to run the test.	FI-UN-31
FI-PR.13	FI-Session.5	Coordination of test execution.	FI-Platform MUST provide means to coordinate FI-Session for certain test suites – e.g. those used for running an interoperability test session between two remote FI-Users or remote FI-User running an	FI-UN.06, FI-UN.21, FI-UN.24

			interoperability session against an automated IUT. The coordination MAY be done via the exchange of messages with the FI-Users describing the actions needed from them. These messages exchanged must be based on the tests descriptions of the test cases. FI-Platform MAY provide an interface for displaying these messages, and MAY provide a services for sending messages between FI-Users of the FI-Session.	
FI-PR.14	FI-Session.5	Control test execution.	FI-Platform MUST provide means for FI-Users to control the test execution for actions such as: start test suite, stop, abort, test case finished pass to next test case, skip test case, restart test case, restart test suite. FI-Platform MUST provide the FI-User with information regarding the state of the automated test execution.	FI-UN.20, FI-UN.22, FI-UN.23,
FI-PR.15	FI-Session.4	Resource provisioning and session setup.	FI-Platform MUST provide FI-User and FI-Contributor with both virtual and physical resources for deploying his/her IUT for testing their implementations in the testbeds. Examples: different OS based VMs, different type of sensor nodes, most common libraries used, etc. Deployed nodes in testbeds MUST be accessible via a secure tunnel (such as SSH).	FI-UN.19,
FI-PR.16	Automated IUT contribution	Upload an automated IUT contribution	FI-Platform MUST provide means for FI-Contributor to contribute with an automated IUT to FI-Platform. As these automated IUT contributions MUST be able to be driven/run by external/remote calls, F-Interop MUST define simple interfaces between the FI-Platform and the automated IUTs adapted to capabilities of the IUT. F-Interop MUST define remote call procedures/actions such as: execution of test case, execution of test suite, IUT configurations. Different levels of automation for the IUT MAY be defined, each one of them containing a certain set of procedures/actions to be implemented by the automated IUT.	FI-CN.06

FI-PR.17	FI-Session.5	Visualization of real-time information during execution of test suite.	FI-Platform MAY provide FI-User with a visualization tool showing relevant network traffic for the test suite in real time, and it MAY provide test results after each test case execution.	FI-UN.25, FI-UN.26
FI-PR.18	FI-Session.5	Access of test logs	FI-Platform MUST provide access to the data exchanged on the encrypted channel between the IUT and the other resources.	
FI-PR.19	FI-Session.6	Results analysis and report	FI-Platform MUST be able to provide the power consumption estimation plugin with the statistics required to perform the energy efficiency estimation.	FI-UN.26, FI-CN.11
FI-PR.20	Test tools	Contribute with a testing tool	FI-Platform MUST provide means to FI-Contributor to contribute with a testing tool to the platform. FI-Platform MUST define interfaces for the integration with the testing tools. Testing tools MUST have references to the authors/contributors.	FI-CN.07
FI-PR.21	F-Interop contributions	View and suggest modifications to source code of test tools.	FI-Platform SHOULD provide mechanisms to allow FI-Contributor to contribute with modifications to other FI-Contributor's test tools. Referenced authors/contributors SHOULD accept those changes before they are deployed in the FI-Platform.	FI-CN.04
FI-PR.22	F-Interop contributions	View, create, modify, delete features for test implementations, with versioning features.	FI-Platform MUST provide the means to view, create, modify and delete test implementations, with versioning features. Test implementations MUST be developed for a previously defined test description and for a particular test tool, so they MUST have a reference to a test description TD-id (TD-id must identify a particular version of the test description) and to a testing tool TT-id (same principle as TD-id). The FI-Platform MAY provide templates and models when generating these new scripts. References to authors/contributors MUST be kept in these scripts. F-Interop-Platform **should** provide mechanisms to allow FI-Contributor to contribute with modifications to other FI-Contributor's test implementations.	FI-CN.05, FI-CN.08

FI-PR.23	FI-Session.7	Results analysis and report generation.	After execution of the test suit, FI-Platform MUST provide a full report of occurrences of each test case, along with their corresponding verdict, provided with explanations in case of FAIL verdicts or encountered problems during execution – e.g. execution errors -. FI-Platform MAY provide means to export a report of the tests results of the executed FI-Session.	FI-UN.28,
FI-PR.24	FI-Session.7	Results analysis and report generation.	FI-Platform MAY provide an interface to display result statistics to authorized FI-User/FI-Contributor.	
FI-PR.25	Certification/labelling program	Inform FI-User about certification/labelling program. Issue of certification label.	FI-Platform SHOULD provide the information needed for the FI-User to apply for certification/labelling program. FI-Platform SHOULD provide means for issuing a certification label after passing a predefined certification test suite; this label MUST be attached to the unique device identity and stored on an online catalogue and accessible. FI-Platform MAY be able to provide a report of some relevant FI-Sessions results proving the device/implementation complies with the requirements for certification/labelling.	FI-UN.29
FI-PR.26	FI-Session.7	Session and results storage	FI-Platform MUST provide means to store information related to the execution of the FI-Session such a session-id, test suite run, test tools used, IUT used, network traffic captures, test results, and any information needed for future re-execution of FI-Session or future re-execution of result analysis.	FI-UN.34
FI-PR.27	FI-Session.7	View list of stored sessions, re-execution of session and re-execution of result analysis with different test implementations.	FI-Platform MUST provide means to list of all previously stored session to the FI-User. FI-Platform MAY provide means to let FI-User re-execute a session. FI-Platform MAY provide means to re-execute results analysis with different test tool implementations or different test tool from the ones previously used if the test tool and analysis methodology allows it.	FI-UN.26, FI-UN.35

Table 5 - F-Interop-Platform requirements

6 References

[1] IETF. [Online]. Available: https://www.ietf.org/rfc/rfc2119.txt.

[2] IETF. [Online]. Available: https://tools.ietf.org/html/rfc7252.

[3] IETF. [Online]. Available: https://tools.ietf.org/html/rfc7641.

[4] IETF Work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-block-19.

[5] IETF. [Online]. Available: https://tools.ietf.org/html/rfc6690.

[6] "6tisch charters," [Online]. Available: http://tools.ietf.org/wg/6tisch/charters.

[7] NMTFtools. [Online]. Available: http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html.

[8] [Online]. Available: http://www.nephos6.com/.

[9] N. L. V. Adrichem, C. Doerr and F. Kuipers, ""Opennetmon: Network monitoring in openflow software-defined networks,"," *Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014, pp. 1–8..*

[10] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang and H. V. Madhyastha, "Software-defined latency monitoring in data center networks," *Passive and Active Measurement. Springer, 2015, pp. 360--372..*

[11] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang and H. V. Madhyastha, ""Flowsense: Monitoring network utilization with zero measurement cost,"," *Passive and Active Measurement. Springer, 2013, pp. 31–41..*

[12] Z. Su, T. Wang, Y. Xia and M. Hamdi, "Flowcover: Low-cost flow monitoring scheme in software defined networks," *Global Commu-nications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 1956–1961..*

[13] G. Gheorghe, T. Avanesov, M. R. Palattella, T. Engel and C. Popoviciu, "SDN-RADAR: Network troubleshooting combining user experience and SDN capabilities,," *Proc. of IEEE Conf. on Network Softwarization (NetSoft), April 2015, London, UK..*

[14] iMinds, "D1.3.1 - Initial architecture design".

7 Annex

7.1 An example of “IPv6 Ready logo” Conformance test description for 6LoWPAN

Test 6LoWPAN.3.2.1: Forbidden action

Purpose: Verify that a router don't send the L bit when sending a router advertisement.

References:

- [6LP-ND] – Section 6.1 (Forbidden action).

Resource Requirements:

- Packet generator
- Monitor to capture packets

Test Setup: The network is setup according to Common Topology. Common Test Setup 1.4 is performed at the beginning of this test part. The Common Test Cleanup procedure is performed after each part.

Router Solicitation A

6LoWPAN Header
IPv6 Header Next Header: 58 Source: NUT’s Link-Local Address
Router Solicitation
Source Link-Layer Address Option

Router Advertisement A

6LoWPAN Header
IPv6 Header Next Header: 58
Router Advertisement
Prefix Information Option L Bit: 0
Source Link-Layer Option

Procedure:

1. The node-1 sends a Router Solicitation (RS).
2. Observe the NUT.

Observable Results:

Step 2: Upon the receipt of Router Solicitation A, the NUT sends Router Advertisement A with the following details:

- the on-link (L) bit unset (0).

Possible Problems:

- None.

7.2 An example of ETSI interoperability test description for CoAP

Interoperability Test Description			
Identifier:	TD_COAP_CORE_05		
Objective:	Perform GET transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.2, 5.8.1		
Pre-test conditions:	• Server offers a /**test** resource that handles GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is requested to send a GET request with: • Type = 1(NON) • Code = 1(GET)
	2	check (CON)	Sent request contains Type value indicating 1 and Code value indicating 1
	3	check (CON)	Server sends response containing: • Type = 1(NON) • Code= 69(2.05 Content) • Content type option
	4	verify (IOP)	Client displays the received information