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

Deliverable D3.4
Spatial and map representation of experiments 1st iteration

Grant Agreement number:	**687884**
Project acronym:	**F-Interop**
Project title:	**FIRE+ online interoperability and performance test tools to support emerging technologies from research to standardization and market launch** **The standards and innovations accelerating tool**
Type of action:	**Research and Innovation Action (RIA)**
Project website address:	**www.finterop.eu**
Due date of deliverable:	**31.10.2017**

Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document properties

Responsible partner	Device Gateway SA (DG)
Author(s)/editor(s)	Eunah Kim (DG), Matteo Filipponi (DG), Michael Hazan (DG), Adriano Perlini (DG), Luca Lamorte (UL), Cedric Crettaz (MI), Sébastien Ziegler (MI)
Version	1
Keywords	F-Interop, interoperability test, performance test, federated networks, Grafana, 2D map, Network Visualization

Abstract

D3.4 is a software type of deliverable, and this document is voluntarily made to provide the overview of the tools developed under T3.3 on F-Interop spatial and map representation tool. In addition, it describes visualization support for WP3 testing tools: a Grafana based data set visualization tool for performance testing tool and the privacy analysis visualization for privacy tool.

This deliverable includes the concepts and general mechanisms of the visualization tools and provides implementation challenges so that the further developers understand the implementation choices and helps on the right decision on the similar work. It also includes message flows of functional modules and snapshots of the represented visualization of each tool.

Table of Contents

Table of Contents **3**

List of Figures **4**

List of Tables **5**

List of Acronyms **6**

1 Introduction **7**

1.1 About F-Interop **7**

1.2 Deliverable Objectives **7**

1.2.1 Work package Objectives 7

1.2.2 Task Objectives 7

1.2.3 Deliverable Objectives and Methodology 7

2 General design framework **8**

2.1 Concept and the general architectural view **9**

2.2 Spatial and map representation tool **11**

2.2.1 Functional components 11

2.2.2 Implementation decision and challenges 12

2.2.3 Message flows 12

2.2.4 Functionalities with snapshots 14

2.3 Data set visualization for performance testing tool **16**

2.3.1 Functional components 16

2.3.2 Implementation decisions and challenges 18

2.3.3 Functionalities with snapshots 19

2.4 Privacy analysis visualization for the privacy testing tool **22**

2.4.1 Concept 22

2.4.2 Visualization 23

3 Conclusion and Future works **24**

List of Figures

Figure 1 Spatial and map representation tool in F-interop architecture 9
Figure 2 Simplified F-Interop remote testing mechanism via Event Bus 10
Figure 3 Server architecture for F-Interop spatial and map representation 11
Figure 4 Message flows for the spatial and map visualization tool 13
Figure 5 Global visualization of all tests in F-Interop 14
Figure 6 Cluster information 14
Figure 7 Single marker zoom-in 15
Figure 8 Individual test status 15
Figure 9 Individual test info 16
Figure 10 Simplified functional components of the data set visualization tool 17
Figure 11 Flow of performance testing tool session with its visualization tool 18
Figure 12 FI-User's Automatic login to Grafana 19
Figure 13 A FI-User starts to see the performance test results 19
Figure 14 Visualization of the packet count and bytes processed 20
Figure 15 Visualization of the transaction RTT and successful transaction 20
Figure 16 Parameters to be visualized (1) 21
Figure 17 Parameters to be visualized (2) 21
Figure 18 Privacy tool execution lifecycle (source: D3.3) 22
Figure 19 Visualization of privacy analysis report 23
Figure 20 A virtual image of the visualization of QoS topology map 24

List of Tables

Table 1 F-Interop Session (Source: F-Interop D1.1)..... 10

List of Acronyms

AMQP	Advanced Message Queuing Protocol
API	Application Program Interface
CoAP	Constrained Application Protocol
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
IUT	Implementation Under Test
NFV	Network Functions Virtualization
QoS	Quality of Service
REST	REpresentation State Transfer
RT	Real Time
SO	Session Orchestrator
SVG	Scalable Vector Graphics
TBaaS	Testbed as a Service
TT	Testing Tool
SDN	Software Defined Networking
Viz-tool	Visualization tool
WP	Work Package

1 Introduction

This section gives general objectives of the F-Interop project, WP3, T3.3 and the deliverable. The deliverable is a software type that was designed and implemented following the design principles and architecture of F-Interop, and integrated into the F-Interop framework with several testing with connecting components.

1.1 About F-Interop

F-Interop is a Horizon 2020 European research project, which proposes to extend the European Future Internet Research and Experimentation (FIRE+) infrastructures with online and remote interoperability and performance test tools supporting emerging IoT-related technologies from research and standardization to market. 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

WP3 is for research and development of performance test tools, a tool for privacy risk assessment, and a spatial representation tool to support experimenters. It includes research and integration of testing tools with network virtualization technologies such as OpenFlow / OpenDayLight based SDN / NFV environments.

1.2.2 Task Objectives

T3.3 is to research state-of-the-art technologies for enabling a flexible visual representation of experiments and to provide a visualization tool supporting performance and privacy tests. The tool enables the user to get a clear representation of the network configuration and node deployment used in the test, both on the user side and on the testbed side. It ensures that the developing visualization tool fulfils scalability requirements and eases the integration of the experiment parameters such as Quality of Service (latency, packet loss, etc.). The task started by specifying the requirements in terms of experiment visualization in conjunction with the other work packages. It then analysed the state of the art in terms of open source solutions in order to compare and select the most relevant one. It customizes and extends them to address the specific F-Interop requirements by following an iterative methodology through several iterations.

1.2.3 Deliverable Objectives and Methodology

This deliverable is to demonstrate the F-Interop spatial and map representation tool and visualization of the Performance and Privacy testing tools. The deliverable D3.4 is not a report, but categorized as “Other” delivered as an initial service provided to the F-Interop platform to enable various forms of visualization of test process and test results. This document is voluntarily made to provide overview of the visualization tools. It includes the message flows of the related components, the APIs, and the snapshots of the visual representation.

The following sections describe details of the F-Interop visualization tools.

2 General design framework

As it briefs in the Section 1, the F-Interop spatial and map representation tool provides a global overview of all tests running on F-Interop platform, while there are test-specified visualization tools.

The task T3.3 (Spatial and map representation of experiments) has focused its initial effort on different types of visualization tools. The work started by analysing the F-Interop platform priority needs and requirements, as well as the architectural implications and design of visualization tools integration. One of the challenging elements has been researched was a fine and adequate solution for the visualization tools to be designed as a modular resources that can be used through the RabbitMQ, and integrated in the Graphical User Interface as a resource.

The architectural model adopted by F-Interop gave us the opportunity to address interesting and challenging requirements. It also required to anticipate a dual evolution scheme:

- The visualization tools should be easily used by future testing tools that are not developed yet, including tools developed through the open call;
- The visualization tool service should be able to easily integrate and integrate additional visualization tools to address specific needs and requirements.

The task researched and compared state of the art open source visualization solutions, and started by focussing on two main priorities for the platform:

- Global process visualization tool: Such tool intends to visualize F-Interop ongoing tests with a spatial and map representation of ongoing interactions among the participants illustrated with an interactive 2D map. This tool will have several functions:
    - to provide a tool for the admin of the platform in order to monitor the ongoing processes;
    - to enable interop test managers to monitor in real time the activities of a many to many interop test event;
    - to ease and save time in identifying available pairing of participants for interop tests.
- Test-specified visualization tools: such tool is dedicated to visualize the results of the performance and privacy testing tools. It includes a data set representation tool utilized with Grafana and InfluxDB for performance testing tool, and a privacy analysis result that is integrated into the GUI.

At this iteration, a Grafana based data set visualization tool has been developed for the F-Interop Performance testing tool. The visualization of the F-Interop Privacy testing tool does not have a separate visualization tool but integrated into the per-test instance from the F-Interop GUI. In the later iteration when SDN/NFV QoS tool will be provided, a dedicated visualization tool for the testing tool will be also provided.

2.1 Concept and the general architectural view

The global 2D map visualizes the global use of the F-Interop platform in near real-time. In the view point of the F-Interop architecture as shown in Figure 1 Spatial and map representation tool in F-interop architecture, the spatial and map representation tool collects information of current users and nodes, and communicates with the GUI module for collecting real-time testing information.

While the spatial and map representation tool is testing-tool agnostic, the specific visualization of a testing tool is made per testing tool.

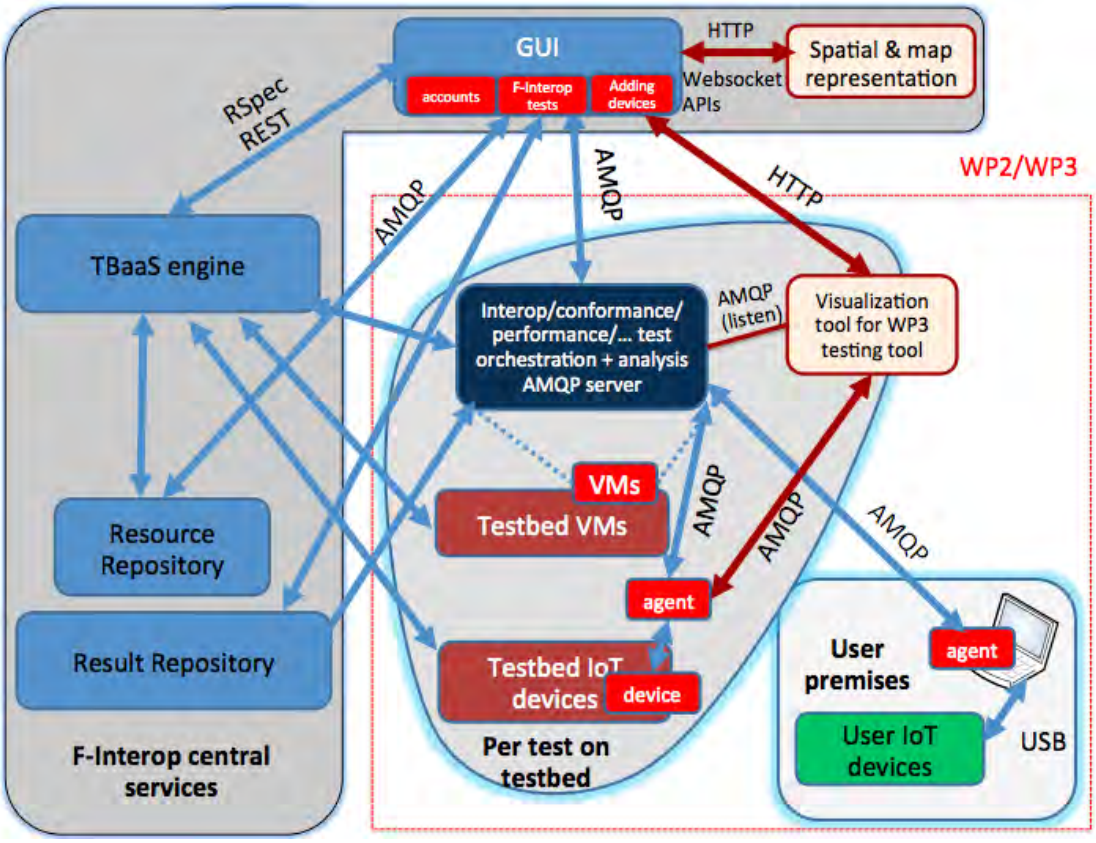

Figure 1 Spatial and map representation tool in F-interop architecture

The F-Interop is composed with different types of testing tools and the central services, and the communication among the components is highly important to keep the flexibility and modularity. AMQP (Advanced Message Queuing Protocol) based event bus has been chosen for message exchanges to support such requirements. This mechanism provides a secure channel for the communication among components including control messages, raw data packets and logs.

As a technological solution to implement AMQP based event bus, F-Interop has adopted RabbitMQ, which provides all capabilities required by the platform as shown in the Figure 2 Simplified F-Interop remote testing mechanism via Event Bus. It is an open source message broker software that implements AMQP and it enables differentiated ACL (Access Control List) security/authorization profiles. RabbitMQ acts as a secure message broker among all components connected through encrypted channels. Each AMQP message exchanged contains a routing key and a topic which indicate how to route that message to the relevant input queues of the component.

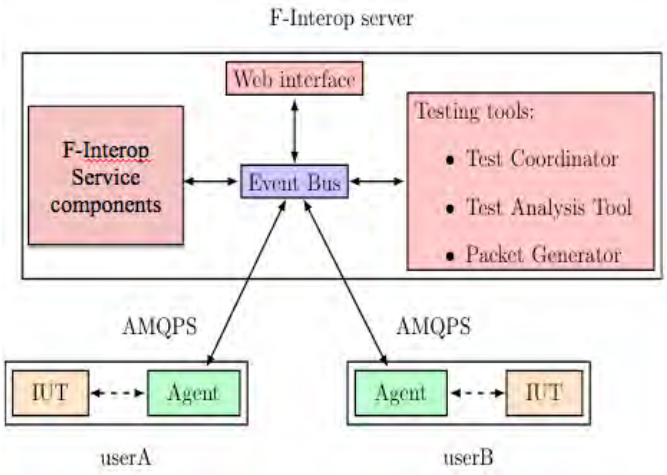

Figure 2 Simplified F-Interop remote testing mechanism via Event Bus

The visualization tool server also uses RabbitMQ for AMQP communication. RabbitMQ consumers are used to retrieve real-time information directly from the session-specific F-Interop virtual hosts as some of this type of information may not be available via the WebSocket interface of the GUI component.

F-Interop deliverable D1.1 summarizes 7 steps of F-Interop session as shown in the **Error! Reference source not found.**. The globally used spatial and map representation tool is visualizing the step 5 (Test execution), and the test-specified visualization tools are visualizing the step 6 (Result analysis and report).

Table 1 F-Interop Session (Source: F-Interop D1.1)

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 analysed. 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 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.

2.2 Spatial and map representation tool

F-Interop spatial and map representation tool represents current use of F-Interop sessions in a 2D world map by displaying the position of the involved resources, information about the experiments and related real-time (RT) events.

2.2.1 Functional components

Figure 3 illustrates a high-level description of the server architecture of the F-Interop spatial and map representation tool. As illustrated in the Figure 3, the server is composed by two main components: the HTTP server and the real-time (RT) controller. The role of the first component is to initialize and provide the content (the visualization tool) to the client after it was requested. The role of the second component is to provide real-time information to the client once the tool has been instantiated on its side.

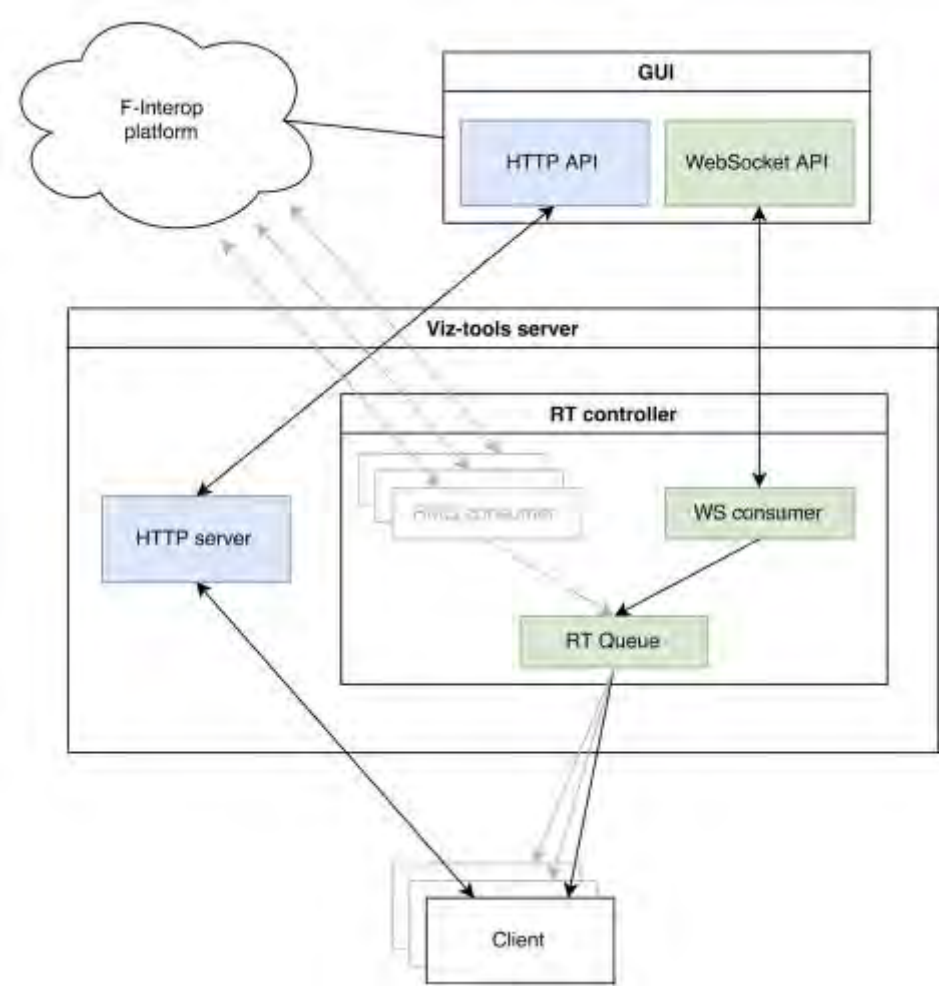

Figure 3 Server architecture for F-Interop spatial and map representation

The RT controller uses the RT queue to buffer all the messages generated from real-time events. The messages are then forwarded to all clients that are currently running the tool and therefore listening to events. At the current state of implementation all the real-time notifications come from the WebSocket API of the GUI component that follows the publish/subscribe pattern. The role of the WS consumer is to use this API and feed the queue with messages. Optionally, when a new session is started, the RT controller can instantiate a RMQ consumer, which will listen to the messages published in the AMQP bus of that session and feed the RT queue with them. This feature was designed to overcome the potential situation where some of the needed data is not available in the WebSocket API of the GUI component (notably exchanged packets between the devices involved in experiments).

2.2.2 Implementation decision and challenges

The server hosting this tool is based on Tornado, which is a web application framework written in Python. Our choice is motivated by many reasons. Firstly, the non-blocking (asynchronous) network I/O gives the possibility to scale up to tens of thousands of open connections. Secondly, the framework supports WebSocket protocol that we need to use to establish long-lived connections with the GUI component and the end-users, so as to be able to display real-time information. Thirdly, the GUI component is based on the same framework, which bring to us maximum compatibility. Fourthly, Tornado is open source and it has a large and vibrant community.

One important challenge from a back-end perspective was the integration of RabbitMQ consumers in the server architecture. This was requested to be able to directly intercept the messages that are exchanged inside a session-specific RabbitMQ virtual host. As a consequence we need to keep an always-up-to-date data structure containing the consumers that is synchronized with the state of the GUI session management, which implies handling correctly the start and stop phases of the consumers. The Python library that implements RabbitMQ (pika) provides an adapter for the Tornado asynchronous I/O loop and this facilitated the integration. At the current state of implementation, these consumers have been tested but not yet included in the production environment.

Another important aspect was to handle concurrent access of the RT queue data structure, this challenge has been facilitated by the polling mechanism of the Tornado I/O loop.

Another challenge was the integration with the GUI component. Due to many architectural modifications, the WebSocket API of the GUI was not available initially and therefore we had to simulate it by implementing our own WebSocket interface. It was also very difficult to test integration because the GUI component is not deployable locally and the documentation of the APIs is not yet complete.

The visualization tool at client side is based on the Google Map (GM) service and the JavaScript library D3. GM is a lightweight service that allows displaying geographical layers and provides extended services such as the Google StreetView API that should give interesting insights in further development that could improve user experience. D3 is a must when it comes to data visualization as it perfectly scales on large projects that require flexibility and performance. It is the most lightest and powerful solution out there. The framework provides every conceivable feature and will easily fit any future requirements. D3 is an open source project and both are used by a large community.

One important challenge in the front-end regarding the development phase was to operate D3 dynamic SVG overlays and Google Map API events and rendering. While the GM API documentation was sometimes weak, we had to overcome issues implementing the overlays, offsetting the projection on D3 to disable unwanted truncated SVG based on the current status of the zoom. Google Map renders and only updates information during the zoom and sometimes the pan events (drag event). Then we had to implement our own clustering regarding overlapping markers depending on the zoom level. We integrated the logical grid-based clustering technique to display hexagon shapes to group those overlapping markers. During the implementation we also had to deal with the case scenario where many markers would share the same location. As many users should be able to start a session in a same location or even interact in that same location, we had to find a way to distribute the positioning of those markers around the city coordinates and expand those markers when a user click on it.

2.2.3 Message flows

The Figure 4 describes the flow of the messages exchanged between the involved components when a user requests the 2D map tool. The main steps are:

(1) The user requests the tool from the viz-tools server with an HTTP GET.
(2) The server requests the currently opened sessions from the REST API of the GUI component.
(3) GUI sends acknowledgement.
(4) The server returns the tool to the user with the necessary information extracted from the open sessions.
(5) The tool is initialized with the received information.
(6) The tool initiates a WebSocket communication with the server. From now the tool will simply listen for RT events.
(7) The server, in particular the RT controller, registers the WebSocket as a listener. Every listener will be notified for a RT event.
(8) An event occurred and the GUI notifies the server through the server-GUI WebSocket communication. Optionally, as previously described, an event can also be notified by a RMQ consumer that is directly connected to a session-specific RabbitMQ virtual host (shown in grey).
(9) The server notifies the event to all listeners (all the connected users).
(10) Depending on the nature of the event, either the method testStarted, testStopped or testEvent is invoked.

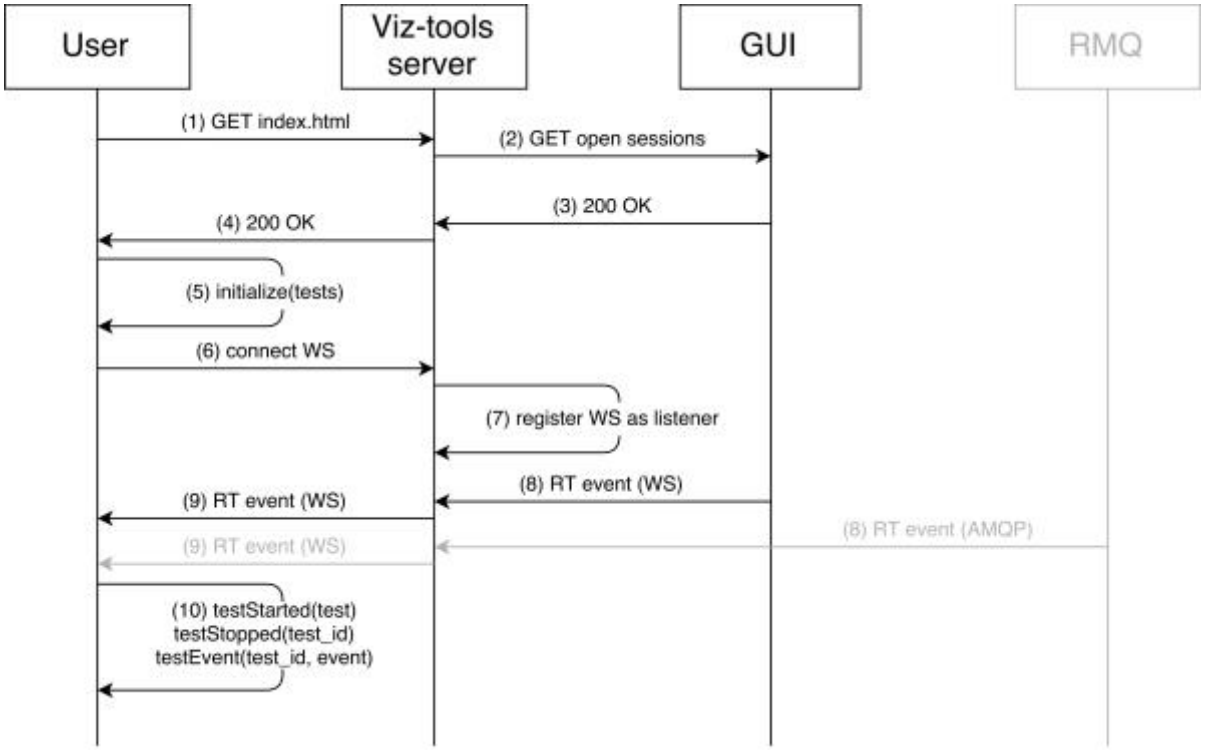

Figure 4 Message flows for the spatial and map visualization tool

2.2.4 Functionalities with snapshots

As shown in the Figure 5, the user sees a global visualization of all the tests being run on a map. There is a sidebar that updates the state of all the running tests, their types and since when they started. The user is able to interact both with the map and the sidebar.

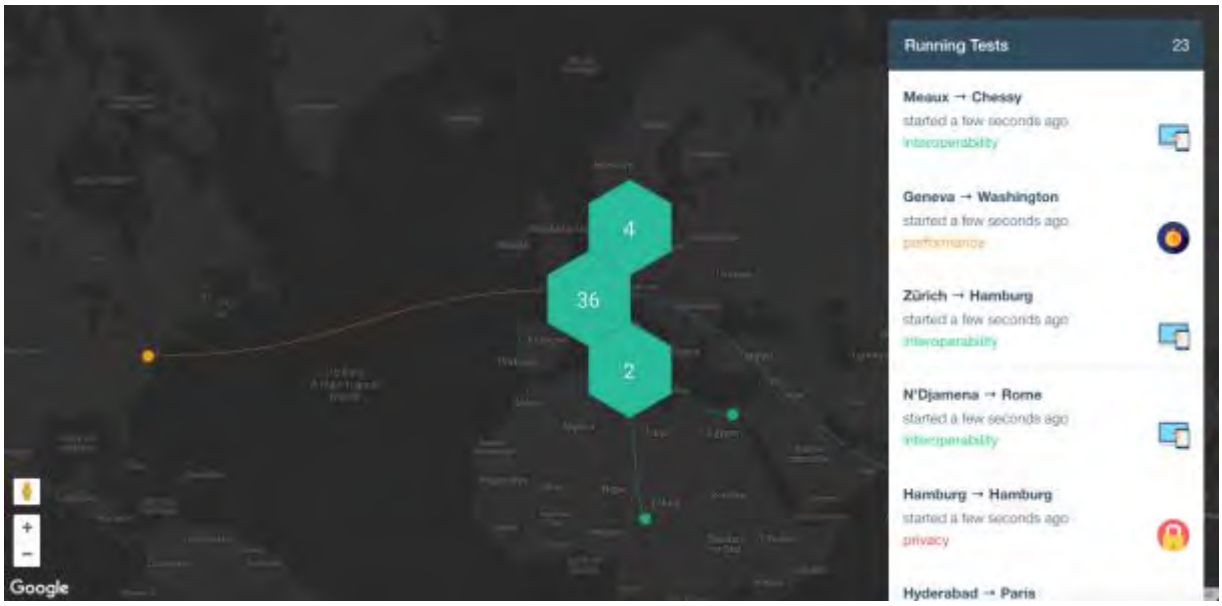

Figure 5 Global visualization of all tests in F-Interop

The user can over his mouse on the clusters to get information detailing the numbers of tests being run and their types as shown in the Figure 6.

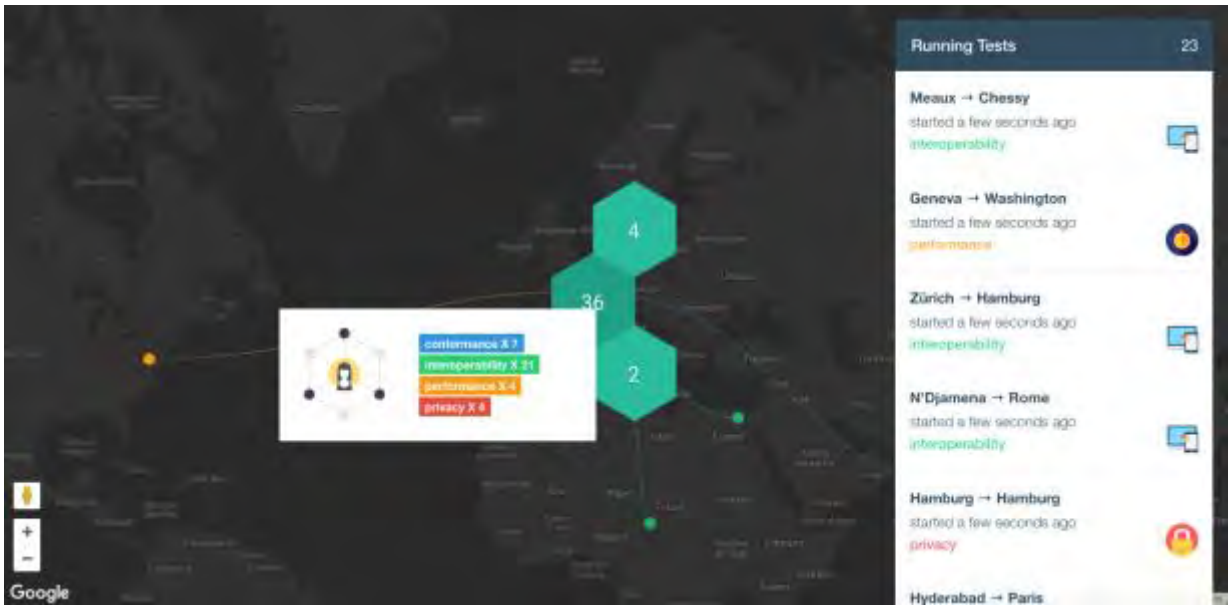

Figure 6 Cluster information

If the user over his mouse on a single marker, he gets the city name, the type of test and since when the test has started as shown in the Figure 7.

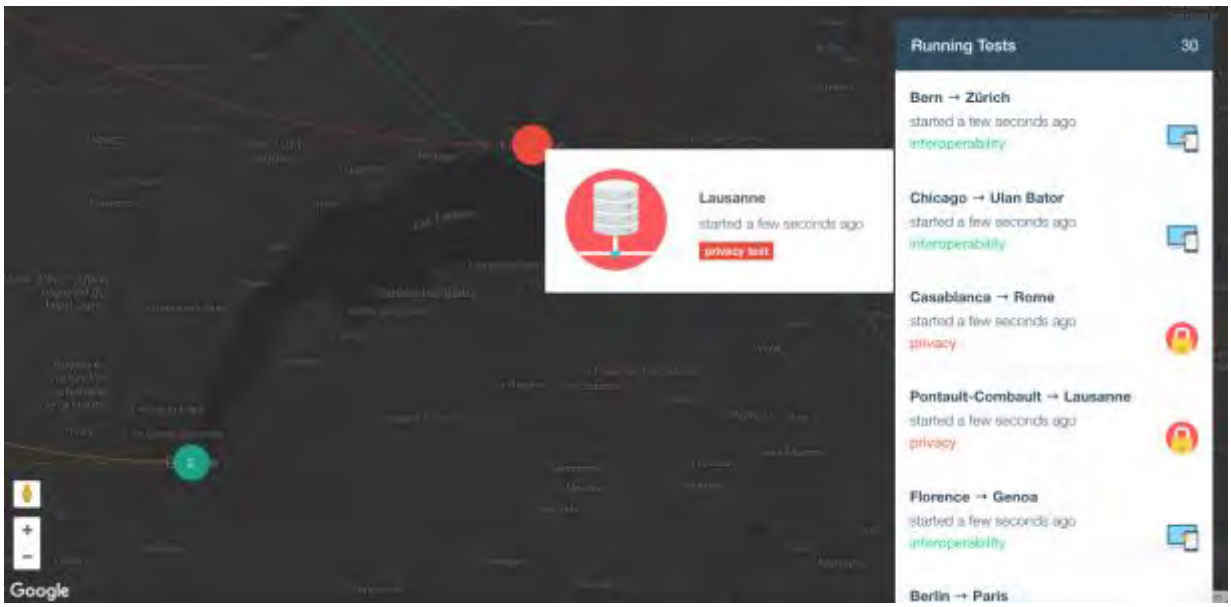

Figure 7 Single marker zoom-in

When the user clicks on a single marker, he updates the zoom state. If he clicks on a cluster that holds many tests on a same location, it activates the closest zoom level and expands all the markers showing all the tests individually as shown in the Figure 8.

Figure 8 Individual test status

Then he can mouse over to get detailed information on each individual test as shown in the Figure 9.

Figure 9 Individual test info

If the user clicks on a test on the sidebar, the map will centre on the selected test. During the entire experiment, the user is able to drag and zoom in and out on the map.

2.3 Data set visualization for performance testing tool

The F-Interop Performance testing tool produces statistic data as a result of test running, and it is visualizing via the Grafana based data set visualization tool.

2.3.1 Functional components

The Figure 10 depicts simplified functional components of the Grafana-based data set visualization tool for the Performance testing tool. GUI returns login information to the master controller of the data set visualization tool so that all data stored in its InfluxDB is also linked to the right user, and each authenticated user can access only his own Grafana dashboard information. Such authentication and authorization are performed between GUI and the data set visualization tool server, and provides automatic login for the F-Interop user. Its master controller (Python script) listens for 'session.created' messages that trigger the starting of a session-specific AMQP consumer. When the performance tool sends the statistic data as the testing result via the encrypted session channel, the data is written into the InfluxDB and sends it to Grafana. The F-Interop user who launched the test sees the results on Grafana dashboard.

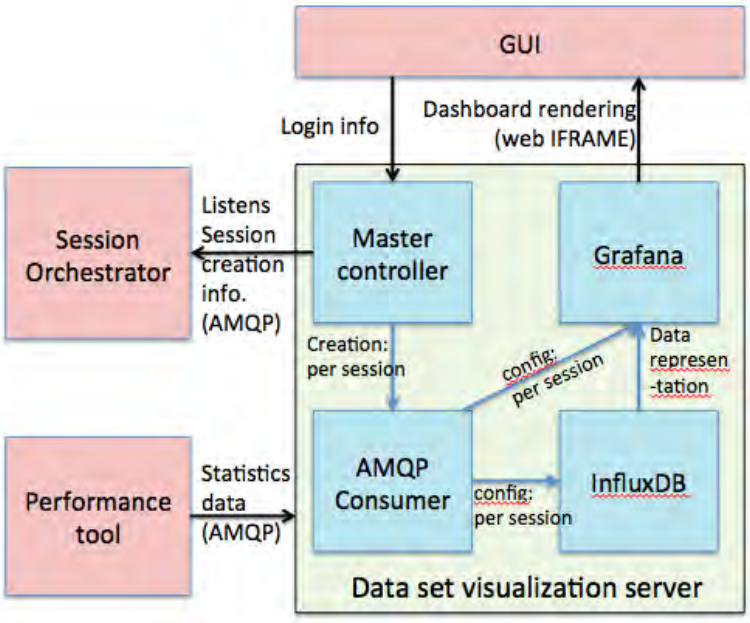

Figure 10 Simplified functional components of the data set visualization tool

The Figure 11 illustrates a session flow of a performance test in FI-Users' perspective. The detailed steps are as followings:

(1) The user is logged into the GUI with its main account and he goes through the test session configuration via GUI.

(2) The GUI tells the SO to deploy a session by using its REST API.

(3) The SO creates and configures an instance of the Testing Tool.

(4) The SO sends a `session.created` message on the `common-services` virtual host and the data set visualization tool controller uses the included session ID to retrieve from the GUI the necessary information to initialize and start the consumer.

(5) The consumer is started and it connects to the virtual host containing the test instance.

(6.a-b) The consumer configures Grafana and InfluxDB entries for the user.

(7) The visualization server needs to send a message to the GUI to confirm that the tool is configured and ready to be used by the user. The message contains a link allowing the user to automatically login in Grafana.

(8) The GUI provides to the user the link, which is embedded in an HTML iframe.

(9) The user connects to Grafana to retrieve the dashboard that will show the test data. The login is automatic (see next section).

(10-11) The session is started by the user.

(12.x) The Testing Tool starts to push the generated data in the session bus and the consumer writes it in InfluxDB.

(13.x) In parallel, the user's dashboard periodically retrieves the new generated data to update the visualization in real time.

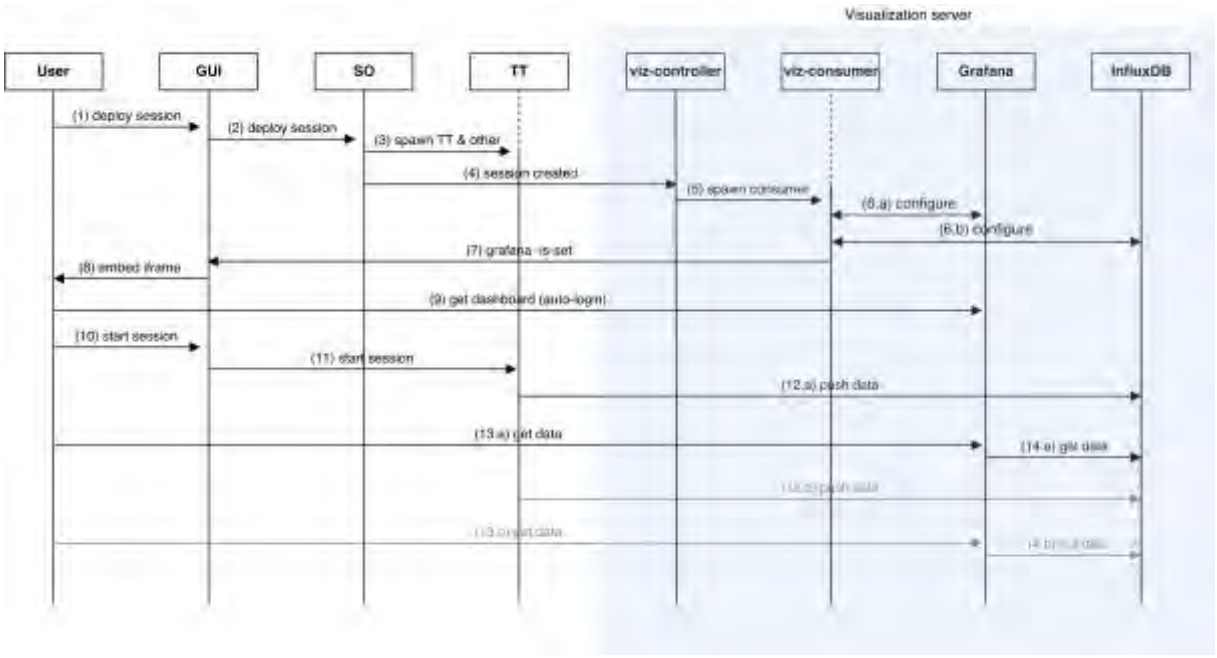

Figure 11 Flow of performance testing tool session with its visualization tool

2.3.2 Implementation decisions and challenges

We explored different available technologies to represent statistic data stored in time series. The two most interesting tools available were Grafana and Splunk, both using InfluxDB for data storage, the choice made, was to use Grafana since Splunk limits the amount of data that can be processed per day in its free version. Grafana also has a larger free contributor base.

We also explored the possibility to run Grafana and InfluxDB in their own Docker container but the choice was made to use an actual server to avoid configuration and debugging issues during the development, this choice was not in line with the project architecture at the time so some adaptations were needed in the performance tool and the SO. The visualization tool needed to be able to get session specific information like the AMQP virtual host and the temporary user login.

The main problem we had to deal with was how to provide automatic login in Grafana. This was necessary to avoid the user entering the temporary username and password to access the dashboard displaying the charts related to the performance test. The goal was that the user should obtain the necessary cookies from Grafana but the problem was that it was not possible to do it programmatically from the browser due to security mechanism. The problem was solved by setting up a reverse proxy between the user and the Grafana server. For each user performing a test, a special secure link is generated that allows the user to connect to the proxy. If the link is valid and not expired, then the proxy will pass the request to Grafana obtaining the cookies, which are then forwarded to the user. Once the user obtained the cookies, all its requests received by the proxy are passed to Grafana without any modification. Figure 12 describes this auto-login mechanism.

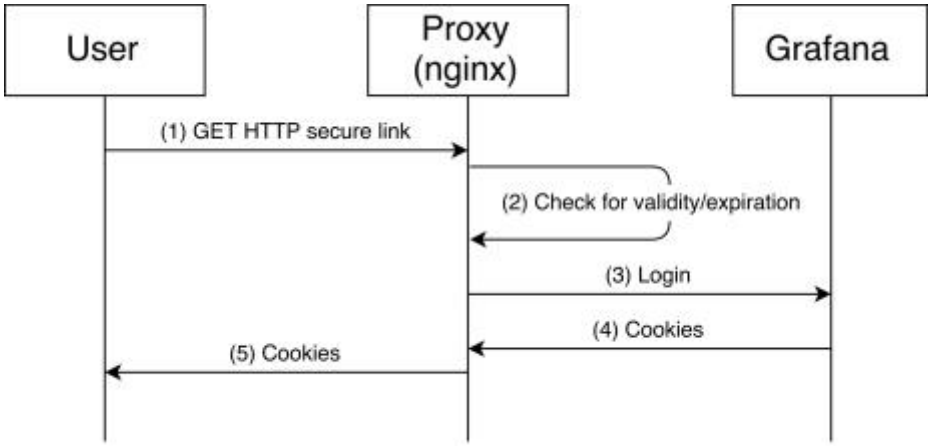

Figure 12 FI-User's Automatic login to Grafana

2.3.3 Functionalities with snapshots

The following snapshots illustrate the integrated Grafana dashboard into GUI, so that the authorized FI-Users see the testing results following the timelines of the test. Figure 13 shows that a FI-User starts to see the testing results from the Grafana dashboard in the page the test ran. The processed bytes are quickly increasing on the timeline.

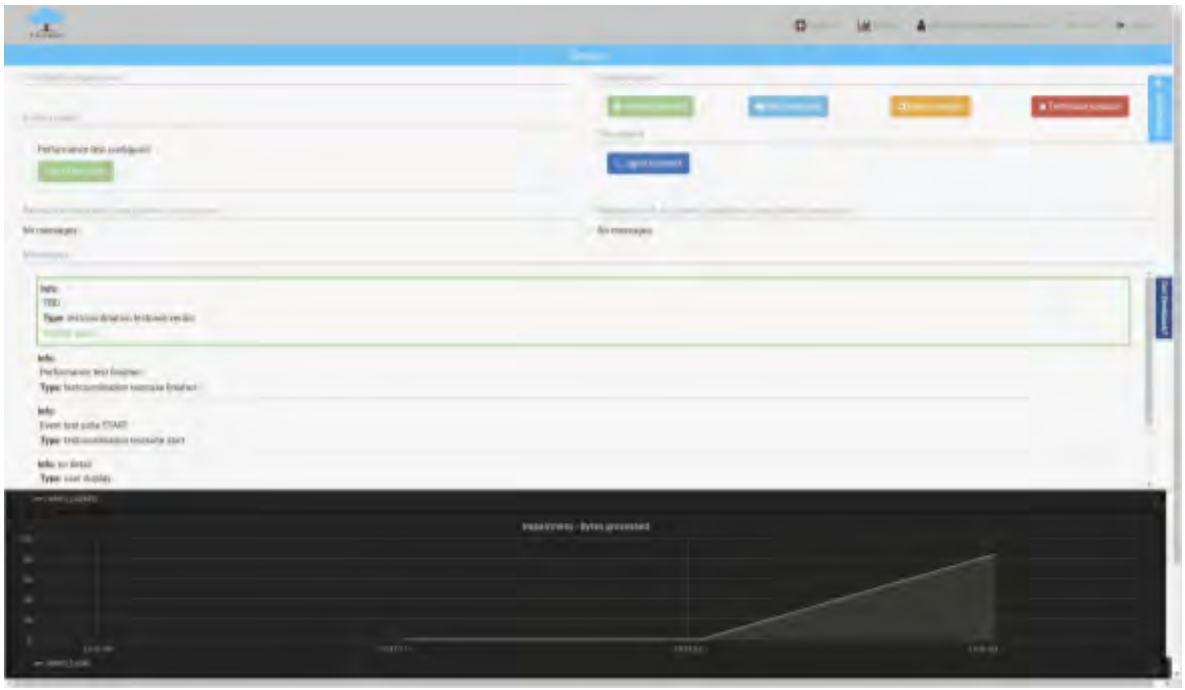

Figure 13 A FI-User starts to see the performance test results

Figure 14 shows that packet counts compared to the bytes processed and a user clicks a certain point on the graph, it shows the value and information on the point and Figure 15 depicts on visualization of the transaction RTT and successful transaction.

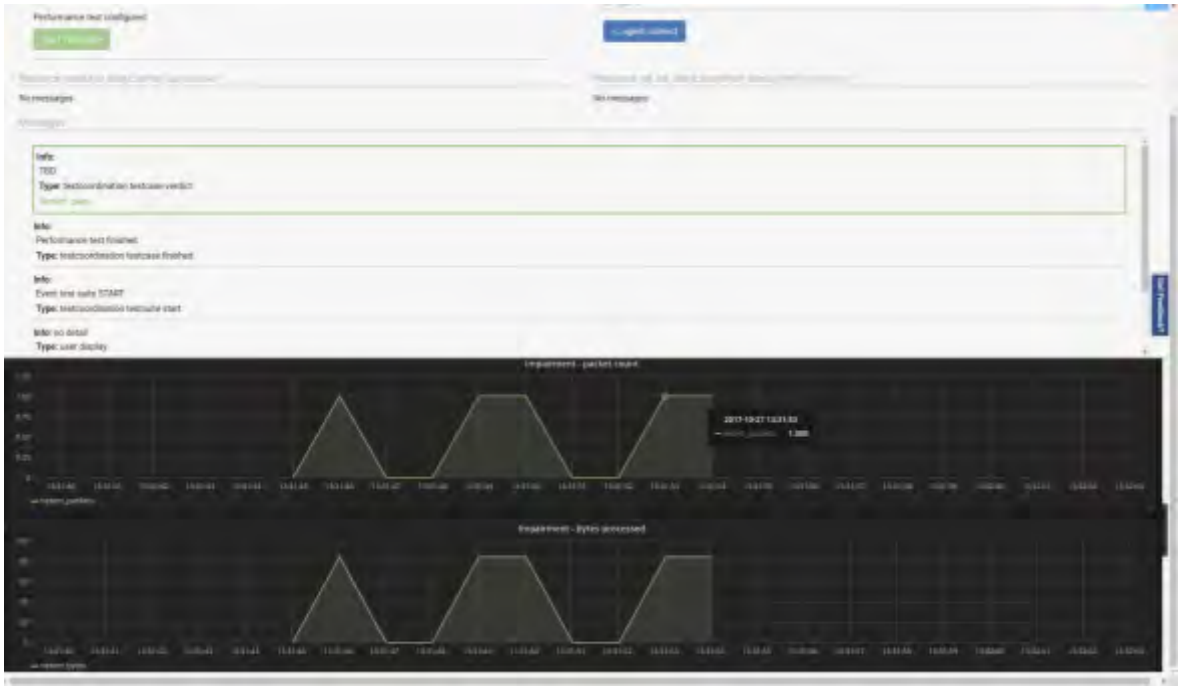

Figure 14 Visualization of the packet count and bytes processed

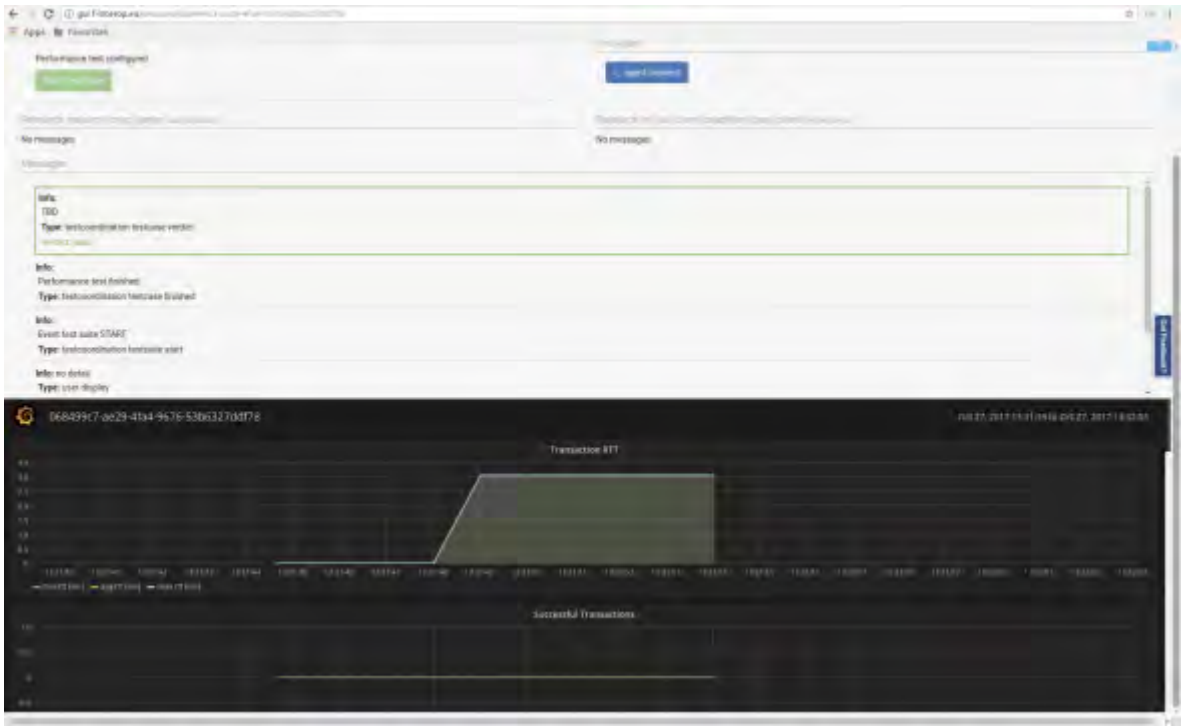

Figure 15 Visualization of the transaction RTT and successful transaction

The user can see ‘attempted transactions’, ‘dropped packet count, ‘overlimit packet count’ and ‘requeued packet count’ as shown in the following.

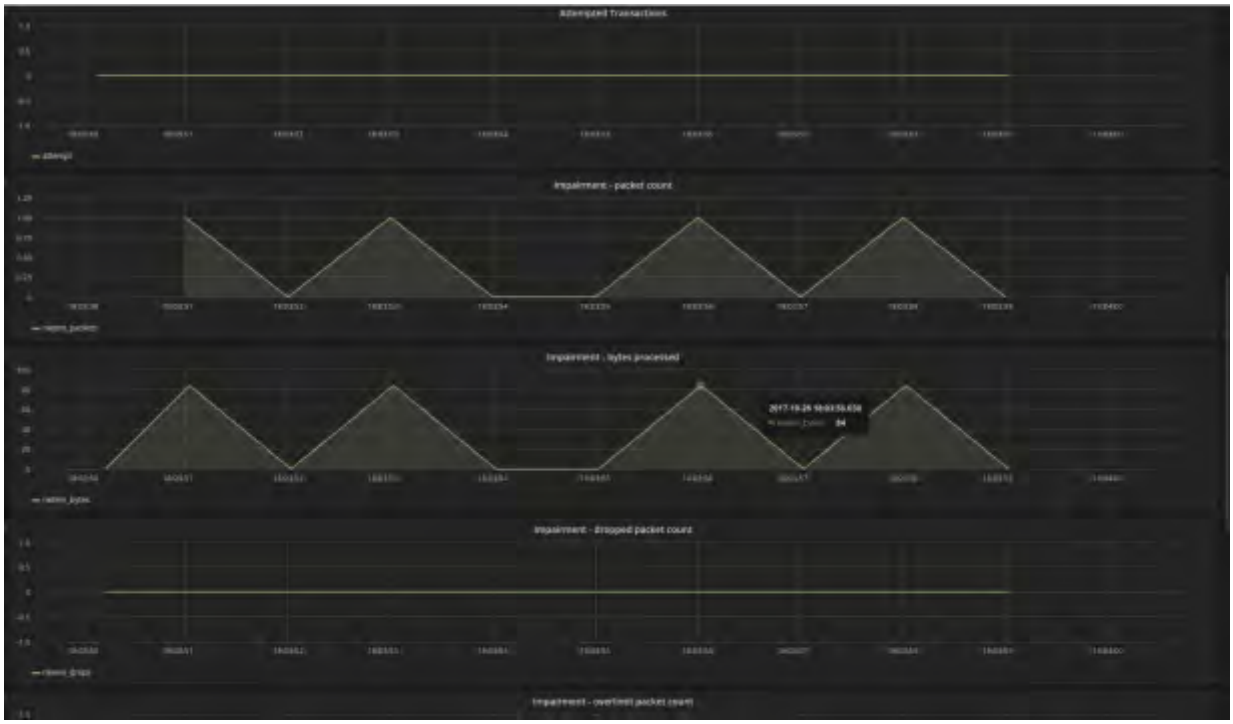

Figure 16 Parameters to be visualized (1)

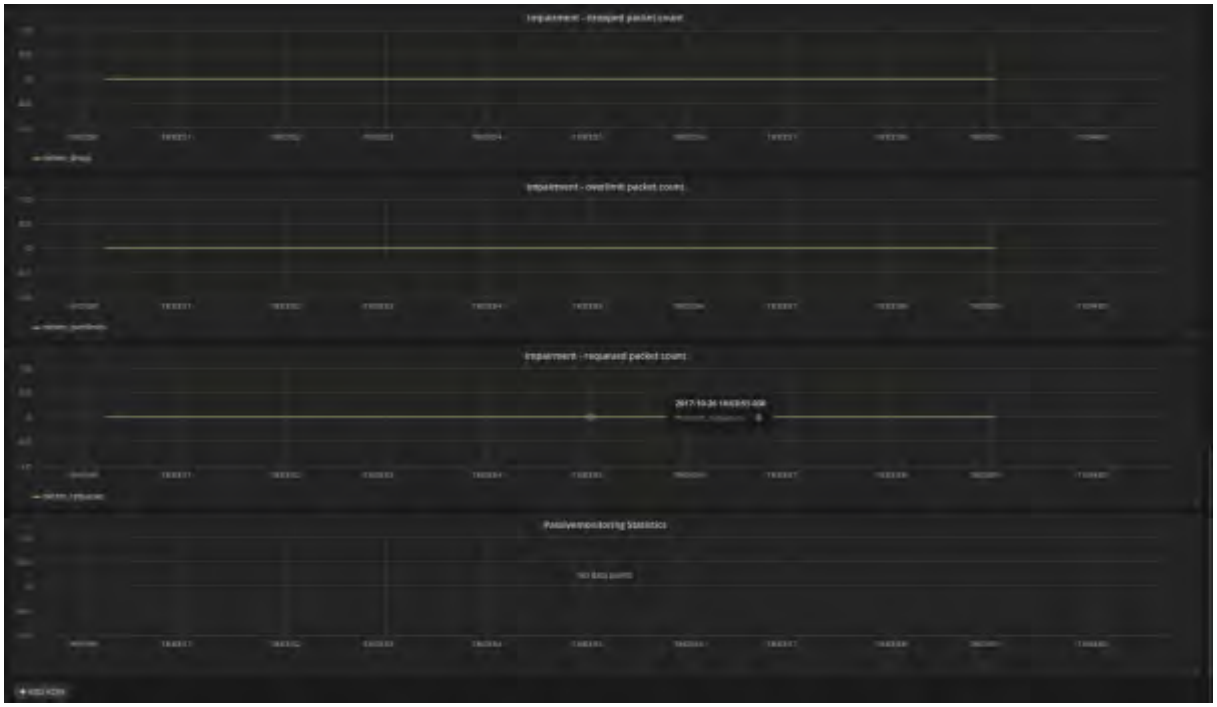

Figure 17 Parameters to be visualized (2)

Grafana automatically refresh the page every 5 seconds and set the windows to the last 10 minutes, this can be changed. The dashboard is predefined in a template located on the Grafana server (in a JSON file) and is loaded on the creation of the user as the following example:

```
{
                "alias":"min rtt [ms]",
                 "dsType":"influxdb",
                 "groupBy":[
                    {
                       "params":[
                          "$interval"
                       ],
                       "type":"time"
                    },
                    {
                       "params":[
                          "null"
                       ],
                       "type":"fill"
                    }
                 ],
```

2.4 Privacy analysis visualization for the privacy testing tool

The Privacy testing tool produces an analysis report after a test has been running. The visualization is included in the GUI showing the privacy concerned information and whether it has been exposed.

2.4.1 Concept

The details of privacy test tool are described in D3.3 that includes the objectives, tool design and specifications of data interfaces through the testing steps. This section briefly describes the visual report of the privacy analysis results on privacy and confidential data leaks while different kind of tests are executed on the F-Interop platform.

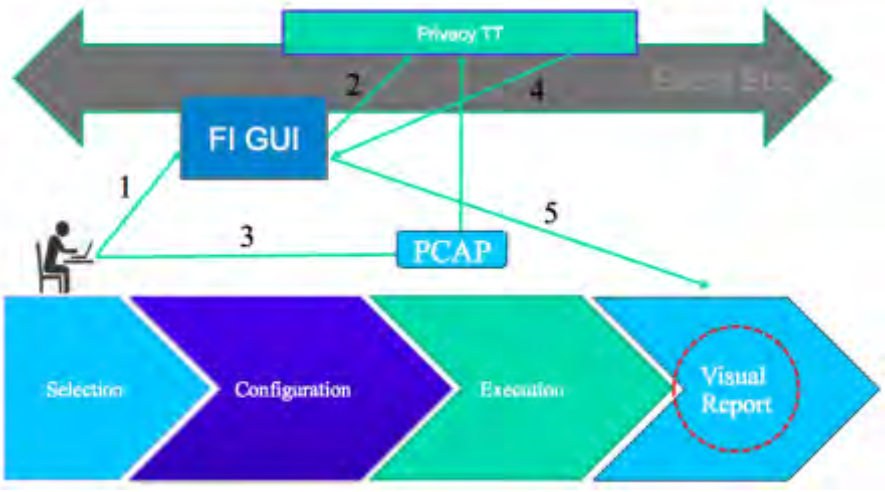

Figure 18 Privacy tool execution lifecycle (source: D3.3)

2.4.2 Visualization

As shown in the Figure 19, the result is delivered to FI GUI via privacy.analysis.reply() in JSON format. The information includes detected issues, frame ID, protocol name, token and privacy tag. The token shows detected privacy information, and the privacy shows categories of the information, e.g., personal info, organization name, etc. Packet payload can be shown to see the details of each detected issue. The Figure 19 illustrated the analysis result after running the tool.

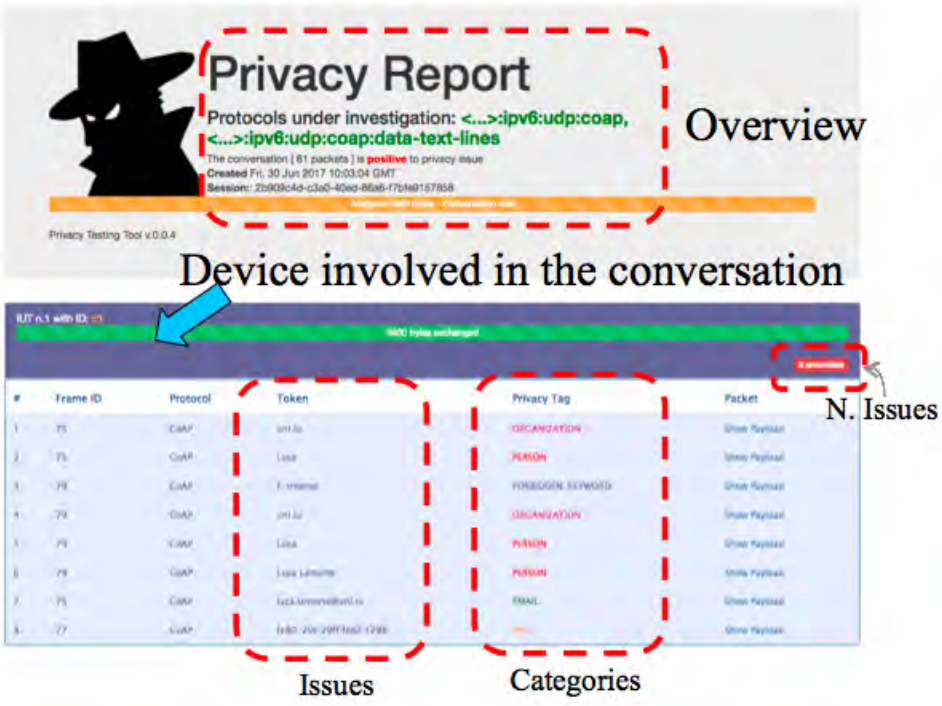

Figure 19 Visualization of privacy analysis report

In order to check the improvement issues of the visualization of the privacy analysis report, it is continuing on discussion with the partner who develops the privacy tool.

3 Conclusion and Future works

T3.3 is responsible on visual representation of diverse F-Interop testing tools including testing tool agnostic spatial and map representation tool, and testing tools specific visualization tools such as Grafana based data set visualization for showing performance testing results and visual report of privacy testing tool.

Some improvement of spatial and map representation tool will be done for the next iteration as followings:

- Continue integration with the GUI component. In particular by improving the compatibility between the visualization tool requirements and the exposed REST API of the GUI, and by implementing the authentication mechanism, this implies dealing with the Cross-origin resource sharing (CORS) mechanism.
- Further collect new visualization requirements in conjunction with the investigation and implementation of new functionalities.
- In order to validate the modularity and scalability of the visualization tools service, 3D visualization tools will be researched. For instance, we consider integrating Streetview to support real time visualization of tests with IoT deployment in smart cities. It can allow a user to see various sensors with their actual position on Street View and the packet being exchanged in the case of a testbed with multiple resources being tested.

The test specific visualization tools may be updated based on the needs from the testing tools. In addition to it, a set of visualisation tools may be added for the testing tools that are currently on development stage.

We are also considering building a visualization tool for the SDN/NFV QoS testing tool in order to represent a network topology with the visualisation of the packet flow and impairment. An image on the discussion is shown in the Figure 20.

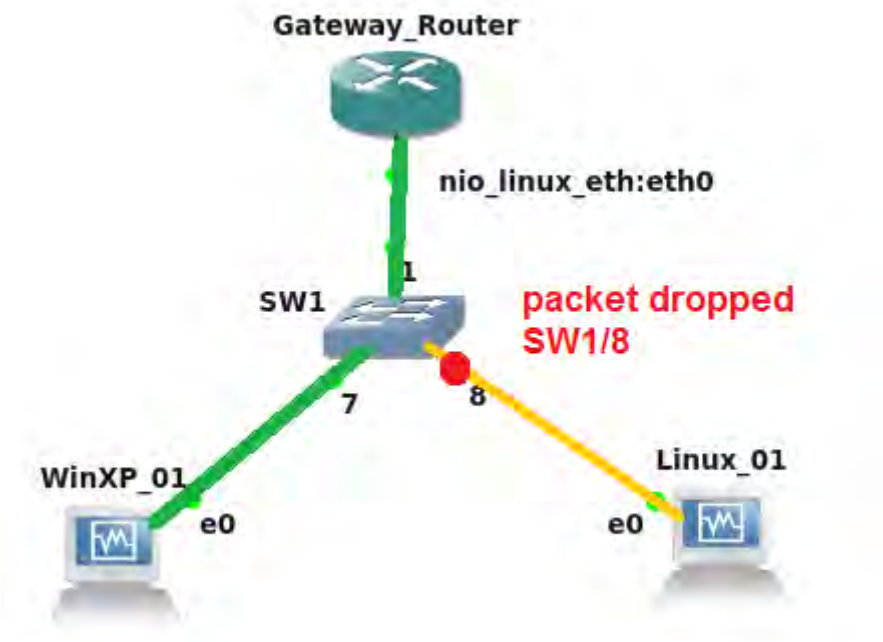

Figure 20 A virtual image of the visualization of QoS topology map