Grant Agreement No.: 732638
Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

FED4FIRE
Federation for FIRE Plus

D3.1: Requirements and specifications for the first cycle

Work package	WP 3
Task	Task 3.1-3.5
Due date	30/06/2018
Submission date	12/11/2018
Deliverable lead	Imec
Version	4
Authors	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI)
Reviewers	Peter Van Daele (imec)

Abstract	This deliverable gives an overview of the requirements for the developments in WP3 of the first 18 months of Fed4FIRE+. WP2 are normal operations developments (add testbeds, fix bugs, small features, etc). WP3 is focussing on larger new functionality.
Keywords	Requirements first cycle, new functionality

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1	1/06/2018	TOC	Brecht Vermeulen (imec)
V2	24/10/2018	First complete version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI)
V3	8/11/2018	Almost final version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI)
V4	12/11/2018	Final version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI)

DISCLAIMER

COPYRIGHT NOTICE

Co-funded by the Horizon 2020 Framework Programme of the European Union

ACKNOWLEDGMENT

This deliverable has been written in the context of a Horizon 2020 European research project, which is co-funded 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.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to FED4FIRE+ project and Commission Services	

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

Co-funded by the Horizon 2020
Framework Programme of the European Union

EXECUTIVE SUMMARY

This deliverable gives an overview of the requirements for the developments in WP3 during the first 18 months of the project. WP2 are normal operations developments (add testbeds, fix bugs, small features, etc). WP3 is focussing on larger new functionality.

WP3 consists out of the following tasks, which are also the sequence of sections in this deliverable:

- Task 3.1 is focussing on SLA and reputation for testbed usage
- Task 3.2 is focussing on Experiment-as-a-Service (EaaS), data retention and reproducibility of experiments
- Task 3.3 is targeting Federation monitoring and interconnectivity
- Task 3.4 works on Service orchestration and brokering
- Task 3.5 researches ontologies for the federation of testbeds

We identify the (user) requirements for the developments in WP3. Detailed requirements for the SLA, Reputation and YourEPM modules are listed. An authentication proxy is identified as a module easing interaction with REST based services.

Besides those, we also looked from a bit further away, to identify needs of experimenters and we found out that Fed4FIRE testbeds do support all kinds of experimentation, but some experiments (e.g. scaling up, NFV/SDN, automation) can benefit from tools doing a lot of the work for the user.

In this regard, we see Fed4FIRE as a meta-testbed where others (e.g. other projects) can build tools on top. Key is then to bring these tools to production quality (with documentation, maturity, etc). D3.2 goes more into detail on some of these tools that have been implemented.

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 SLA AND REPUTATION SERVICE

In Fed4FIRE+ environment, the Service Level Agreement (SLA) and the Reputation Service provide the necessary tools and mechanisms for delivering to the users a quantitative view of the trustworthiness of the federated testbeds. This service facilitates the Fed4FIRE+ users to select the appropriate testbed in the federation according to their experiment's requirements.

The aim of adding SLA within Fed4FIRE+ is to enable testbed providers to create offerings that experimenters can accept establishing an agreement with the testbed owner. We can understand the agreement as a contract between the platform providers and the testbed users. Once the agreement has been created it must be verified that it is being fulfilled. The information related to the execution of an experiment, i.e., if there is an agreement violation, will be send to the other components using a notification / subscription pattern.

The Reputation Service of Fed4FIRE+ aims to enhance and extend the already-developed reputation service of Fed4Fire project. The updated service will leverage Quality of Service (QoS) metrics, such as Availability, Latency etc., Quality of Experience (QoE) metrics, e.g., Usability and Documentation Readability, and SLA data in order to compute the degree of confidence of both experimenters and testbed. At the end of an experiment, the users will be prompted to give their feedback for the reserved testbeds in order to update the reputation score of the testbed and the credibility score of the experimenter. This process mitigates the effect of abnormal or malicious evaluations and guarantees that the testbeds' reputation score is fairly computed.

Co-funded by the Horizon 2020
Framework Programme of the European Union

1.1 SLA AND REPUTATION SERVICE ARCHITECTURE

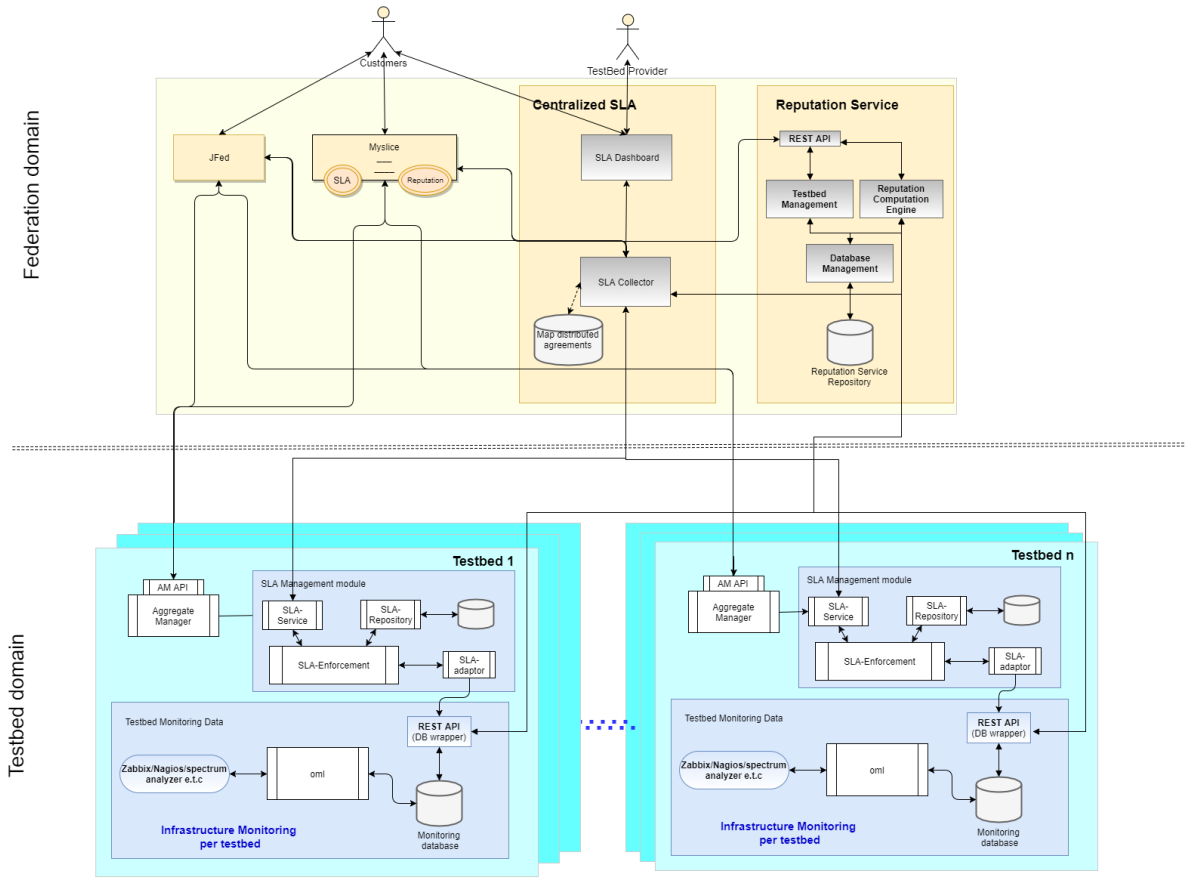

Figure 1: SLA & Reputation Service Architecture

Figure 1 shows the architecture of the SLA and Reputation service within the Fed4FIRE+ project. In the *Federation domain* we have the components that will be centralized and only instantiated once in Fed4FIRE+. In the *Testbed domain* we have the software components that will be installed at each platform and therefore instantiated multiple times in the Fed4FIRE+ project.

The centralized components developed in this task are the SLA dashboard and collector, and the Reputation Service. As shown in the Figure above, both the SLA collector and the Reputation Service communicate with MySlice v2 and jFed tool in order to enable the users to use graphical tools to interact with those components. In the The decentralized components are the SLA management module and the monitoring data databases and APIs. At each testbed, both of them are utilized in order to provide SLA functionalities and integrate with the Reputation Service. The SLA management module and the Reputation Service will access

Co-funded by the Horizon 2020 Framework Programme of the European Union

the monitoring data needed from the API for the calculation of SLA violations and reputation scores.

1.2 FUNCTIONAL REQUIREMENTS

The above architecture and the underlying components should fulfil the functional requirements listed below:

1.2.1 SLA Functional Requirements

ID	SLA_01
Title	**SLA solution must cover the whole lifecycle specified in WS-Agreement**
Short description	The solution must cover the SLA lifecycle: • Generation of WS-Agreement templates and agreements • Provisioning of the agreements and its monitoring. • Management of SLA related entities: templates, agreements, providers, violations and penalties • Assessment of Service Level Objectives (SLOs) and generation of corresponding penalties when an SLO is violated • Notification of detected violations and incurred penalties to the SLA Collector in order to handle it with the subscription service. • Stop the agreements and their monitoring
Additional Information:	A platform owner, but no other, must be able to create an offering. Technically this is implemented by creating an SLA template and including measurable terms. The information of the violations must be generated throughout an experiment lifetime.
Type	FUNC
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_02
Title	**SLA solution REST interface**
Short description	The SLA solution will provide a REST interface in order to enable third-party applications to interact with it. The third-party software must be able to retrieve the details about an offering, template or enforce (start the execution) of an agreement. The result of the execution of an agreement must be also available via the REST interface. The message format must be in XML or JSON.
Additional Information:	-
Type	FUNC / ENV
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_03

Co-funded by the Horizon 2020 Framework Programme of the European Union

Title	**SLA solution Subscription mechanism**
Short description	SLA must provide a subscription mechanism in order to allow third-party software to receive the information of the violations that are occurring in a specific agreement while the agreement is enforced or at the end of the agreement enforcement. The subscription mechanism must enable filtering the messages based on the content.
Additional Information:	-
Type	FUNC
Priority Level	Medium
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_04
Title	**SLA solution multitenant**
Short description	SLA must support the recording of offerings and agreements from different organizations in such a way that the organizations cannot interfere with each other.
Additional Information:	As the architecture is not centralized, every testbed will have different data bases and data between them will not be shared.
Type	FUNC
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_05
Title	**SLA Dashboard**
Short description	The SLA Solution must provide a GUI to simplify the task of testbed providers of creating new offerings and to check the agreements that have been created based on its offerings. Detailed information associated to the offerings and agreements like the terms to be fulfilled or the violations that have occurred must be also identifiable with this GUI. Moreover, the experimenters will be able follow up the agreements created outside the dashboard, since the dashboard is not responsible to create new agreements.
Additional Information:	-
Type	USE
Priority Level	Medium
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_06
Title	**Agreement creation and enactment**
Short description	The agreement creation will be done with jFed or MySlice tools. It must be always based to an offering created by a testbed provider. This agreement

Co-funded by the Horizon 2020 Framework Programme of the European Union

	must always include the terms that must be fulfilled, the expiration time and the id of the offering it is based on. The SLA solution must allow creating an agreement between platform provider (testbed owner) and testbed client (experimenter) based on an offering
Additional Information:	The testbed client must be able to check the existing offering in order to find out if there is an interesting one.
Type	FUNC
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_07
Title	**SLA terms quantizable**
Short description	SLA offerings and agreement will contain terms that must be guaranteed. These terms must be quantizable and comparable.
Additional Information:	This is a functional requirement needed by the SLA. The testbed providers must indicate terms that must be quantizable and comparable, otherwise the SLA solution won't be able to calculate if the terms are being fulfilled or not.
Type	FUNC
Priority Level	Medium
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_08
Title	**SLA access to monitoring data**
Short description	SLA solution must have access to the monitoring data and it must be able to retrieve it using terms that specified by the testbed provider in the guarantee terms. Once an agreement has been created, it must be able to monitor it and calculate if violation occurs or not
Additional Information:	In order to calculate if a violation occurs or not, the SLA solution must access to the monitoring data. The agreement will contain different equation with terms and comparison that must be fulfilled (guarantee term). The monitoring data must be retrieved based on the names of the terms. The values that are retrieved must be comparable with the expression used in the equation to be fulfilled.
Type	DATA
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_09

Co-funded by the Horizon 2020 Framework Programme of the European Union

Title	**Distributed federation architecture.**
Short description	A decision taken in Fed4Fire is that the SLA solution has to be distributed, and this is a requirement that will remain in Fed4FIRE+.
Additional Information:	The SLA solution will allow the integration with other components in order to manage the agreements exposing interfaces.
Type	ENV
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

ID	SLA_10
Title	**SLA solution software dependencies**
Short description	The different SLA components can have different technologies and they must expose REST APIs to communicate each other. They have to be modular and decoupled.
Additional Information:	Technologies used Python and Java based on the results of Fed4Fire.
Type	ENV
Priority Level	High
Identified by Partner(s)	ATOS
Status	Design / Early implementation

1.2.2 Reputation Functional Requirements

ID	REPUTATION_01
Title	**Reputation Service REST interface**
Short description	The reputation service will expose a REST API for the following use cases. The administrators of the reputation service and the federation should be able to add/update/remove testbeds and their services to the reputation service. Also, the experimenters and the frontend tools should be able to retrieve reputation scores of all testbeds and submit new evaluations through the REST API.
Additional information	The message format must be in JSON.
Type	FUNC / ENV
Priority Level	High
Identified by Partner(s)	NTUA
Status	Design / Early implementation

ID	REPUTATION_02
Title	**REST API documentation**
Short description	The APIs endpoints, input formatting and response messages and codes should be thoroughly documented mainly for extensions, usage from other components and integration with the frontend tools

Co-funded by the Horizon 2020 Framework Programme of the European Union

Additional information	-
Type	SUP
Priority Level	Medium
Identified by Partner(s)	NTUA
Status	Design

ID	REPUTATION_03
Title	**Reputation computation engine**
Short description	The new reputation algorithm must be developed, deployed and exposed to the REST API in order to receive evaluations and compute the updated reputation scores of the testbeds.
Additional information	-
Type	FUNC
Priority Level	High
Identified by Partner(s)	NTUA
Status	Design

ID	REPUTATION_04
Title	**Reputation Service access to monitoring and SLA data**
Short description	The new reputation computation engine will access through APIs the monitoring data of an experiment and the information about SLA agreements and violations in order to calculate the user's credibility and readjust the user's evaluation if needed. More specifically, monitoring data and SLA data will compare with the experimenter's evaluation in order to adjust the credibility and the evaluation.
Additional information	The monitoring data will be retrieved from the monitoring data REST API of each testbed while the SLA data will be retrieved through the SLA Collector.
Type	DATA
Priority Level	High
Identified by Partner(s)	NTUA
Status	Design / Early implementation

1.2.3 MySlice Functional Requirements

ID	MYSLICE_01
Title	**Users registration**
Short description	A new user should be able to register from the web frontend, the request for an account must be sent to a manager of the federation.
Additional Information:	
Type	FUNC
Priority Level	High

Co-funded by the Horizon 2020 Framework Programme of the European Union

Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_02
Title	**User account approval**
Short description	A manager of the federation should be able to approve or reject a request from a new user for the creation of an account.
Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_03
Title	**Automated retrieval of Credentials**
Short description	Credentials should be retrieved automatically by the web frontend, hiding the complexity to the user.
Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_04
Title	**List, browse and select resources**
Short description	A user-friendly interface should be presented to the user in order to list, browse and select the relevant resources for his/her experiment (availability, properties, location…).
Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	Ongoing Integration

ID	MYSLICE_05
Title	**Reserve resources**
Short description	The web frontend should forward the list of resources with the time and duration of the reservation requested to the relevant AM within the federation.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_06
Title	**Frontend plugins development guide**
Short description	A documentation explaining how one can develop a plugin in MySlice Frontend using ReactJS framework
Additional Information:	
Type	ENV
Priority Level	High
Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_07
Title	**REST API documentation**
Short description	A documentation describing the REST API of MySlice
Additional Information:	
Type	USE
Priority Level	High
Identified by Partner(s)	UPMC
Status	Implemented

ID	MYSLICE_08
Title	**WebSocket API documentation**
Short description	A documentation describing the WS API of MySlice
Additional Information:	
Type	USE
Priority Level	High
Identified by Partner(s)	UPMC
Status	On-going

ID	MYSLICE_09
Title	**Support of Federation API v2**
Short description	Develop in MySliceLib a client API for the Federation v2 API

Co-funded by the Horizon 2020 Framework Programme of the European Union

Additional Information:	
Type	FUNC
Priority Level	Optional
Identified by Partner(s)	UPMC
Status	On-going

ID	MYSLICE_10
Title	**Deployment an instance of C-BAS**
Short description	Deploy an instance of C-BAS to enable Speaks-for Credentials support
Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	On-going

ID	MYSLICE_11
Title	**Deployment of an instance of MySlice v2**
Short description	Deploy a dedicated instance of MySlice v2 for Fed4FIRE+
Additional Information:	
Type	FUNC
Priority Level	High
Identified by Partner(s)	UPMC
Status	On-going

1.2.4 Type of requirements

Functional	Functional	FUNC
	Data	DATA
Non-functional:	Look and Feel Requirements	L&F
	Usability Requirements	USE
	Performance Requirements	PERF
	Operational - Environmental Requirements	ENV
	Maintainability and Support Requirements	SUP

Co-funded by the Horizon 2020 Framework Programme of the European Union

1.3 SLA

Three components will form the SLA solution that will be described in detail in 1.3.1. These components are *SLA Dashboard* and *SLA Collector* that are located in the *Federation domain* and the *SLA Management module* will be placed in the *Testbed domain.* From the component distribution we foresee that the SLA components will need inter-domain communication, hence the security solution that will be chosen in Fed4FIRE+ for this communication has to be also applied between the *SLA Collector* and the *SLA Management module.* The three mentioned components were already used and distributed in the same manner for Fed4FIRE project, but some changes must be implemented and features must be included in order to adapt to the new architecture and cover the new functionalities.

The Testbed provider will be able to create an offering using the *SLA Dashboard* web-based frontend. Testbed clients have to create the agreement using *jFed* or *MySlice* components. This agreement has to include information such as the Testbed client that has created the agreement, the terms that must be fulfilled, the agreement has an expiration time, the identification of the experiment in which the agreement is based.

The *Aggregate Manager* is the component that is responsible for managing the experiments of the testbeds. It will instantiate and, therefore, it will be the one that will communicate the *SLA Management module* that the agreement has to be started, i.e. the terms from the agreement monitored, evaluated and violations detected.

The *SLA Management* will retrieve the values for the metrics used in the terms in the agreement from the *Testbed Monitoring Data*. A REST API will be created by this component in order to retrieve the monitoring information related to the resources used within a specific experiment. Moreover, the *SLA Management* will evaluate the terms defined and check their fulfillment. If a violation is detected, the information will be, on one hand, stored internally in the *SLA Management* database, and on the other hand, will be forwarded to the *SLA Collector* which will include a notification /subscription service.

Other components will be able to establish which messages they want to receive specifying a filter based on the message content. It will not be a classical subscription where a component receives all messages that are being sent in a queue. Instead, we are going to enable subscription to messages from a specific queue allowing filtering from messages depending on the content of the messages. A formatted message will be sent in this queue and a subscribed component will be able to ask to receive messages only if a specific field is included in the message, a specific value for the field or the value is above or below a threshold. Components like the *Reputation Service* will take advantage of this subscription feature and will be able to optimize their algorithm.

1.3.1 SLA Components & Software/Tools

As we already have introduced in the previous section, three components will compound the SLA solution in the Fed4FIRE+ project. These components are the following:

- **SLA Dashboard:** The *SLA Dashboard* offers a web-based Graphical User Interface (GUI) for testbed providers to create offerings from which the agreements will be

Co-funded by the Horizon 2020 Framework Programme of the European Union

created. These offerings can be then selected by experimenters, when performing the resource reservation, to the one that best matches their needs.
The tool also allows testbed providers to visualize the status of the agreements that have been created using their offering and resources. The dashboard supports agreements in various stages, including definition, production or completion, also indicating whether they have been fulfilled in the latter case. Later, in section 1.6.1 we will include more information about it.

- **SLA Collector**: This is a common element for the federation whose purpose is to act as a central communication point for the client tools and the *SLA Management* module located in each testbed offering SLAs. Client tools only need to communicate with the *SLA Collector* indicating which testbed they want to access and the *SLA Collector* will perform the appropriate calls to the corresponding testbed. This component offers a RESTful API to ease the whole SLA process (from creation to destruction) for client tools.

  The version from Fed4FIRE+ will include the subscription software, the one that will allow other components to receive the violations that are occurring in agreements from the executing experiments filtered on the message content. The *SLA Manager* will always generate the same format of message and it will be this subscription manager that will enable the subscription based on message content.

- **SLA Management module (SLA Core)**: Located at testbed level, this module is in charge of managing offering and agreements (SLAs). It performs the creation, deletion and evaluation of SLAs during the experiment lifetime between the parties involved: experimenters (customers) and testbed owners (providers). The component follows the WS-Agreement (WSAG) specification. This means that the documents representing templates and agreements are valid according to the schema defined in that specification, which could be extended to cover specific Fed4FIRE+ requirements. The core is responsible for managing the actually SLA execution for each federated testbed: i) Generation of WS-Agreement templates and agreements ii) Management of SLA related entities: templates, agreements, providers, violations, penalties iii) Assessment of SLOs and generation of corresponding penalties when an SLO is violated iv) Notification of detected violations and incurred penalties to the SLA Collector in order to handle it with the subscription service. Besides, it includes a RESTful API interface allowing programmatic access to the different types of functionalities offered, and data can be sent in JSON or XML format.

Both, **SLA Dashboard** and **SLA Collector** are based on the same technologies; therefore, they will have same software requirements:

Components	Technologies
SLA Dashboard – SLA collector	Mysql >= 5.0 Python >= 2.7
SLA Manager	Mysql >= 5.0

Co-funded by the Horizon 2020
Framework Programme of the European Union

	Oracle JDK >=1.7

Table 1: SLA technical requirements

Other requirements needed by the SLA Manager at a higher level is that the terms that are included in the offerings and agreements must be something measurable and must be included in the monitoring data. The data values must be numerical and using the same units as they're specified in the offering and/or agreement.

1.4 REPUTATION

The Reputation Service of the Fed4Fire project was based on FTUE reputation framework [1]. For a conducted experiment, this framework used some predefined QoS metrics, i.e., Node Availability, and two QoE metrics, named Overall Experience and Quality, to update the reputation score per testbed per service. Furthermore, the credibility of the users was measured and considered on the final computation of the reputation score. The computation of the experimenter's credibility was based on the difference between the measured QoS metrics and the opinion of the user. The QoS and QoE variables had a specific set of numeric values, i.e., 1-5. Furthermore, no SLA data were used on the computation of the reputation score.

In the context of Fed4FIRE+ and based on the FTUE framework, we will develop a new reputation algorithm with better malicious user filtering. The new Reputation Service will use several QoS metrics, such as Node Availability, Link Availability, and Server Availability, and QoE metrics, e.g. Usability, Document Readability, testbed owner's support, in a hierarchical and scalable structure. These metrics will be expressed by fuzzy variables. Fuzzy logic is suitable to express the nature of QoE metrics. Thus, a set of linguistic values, such as 'Poor', 'Good' and 'Excellent', will be used for the user input and these terms will be mapped to the corresponding fuzzy values. Furthermore, Fed4FIRE+ reputation service will leverage SLA data to accurately update the experimenter's credibility score. This new service will able to provide an overall reputation score per testbed or an individual score per service per testbed.

1.4.1 REPUTATION components & Software/Tools

As presented in Figure 1, the Reputation Service is centralized and implemented as an REST API web-app using the Model-View-Controller (MVC) paradigm. The inner core of the web-app will consist of the testbed management component, and the reputation computation engine, as shown on Figure 1. Both components are in fact a separate Ruby on Rails controller. The database management component consists of the Ruby on Rails models and the Active Record Object-relational mapping (O/RM) tool.

- **Testbed management component:** The testbed management component is addressed to administrators of the Reputation Service. It will be used to add, remove or update testbeds to the Reputation Service. In parallel, it will be responsible for adding, updating or deleting the services of a particular testbed. For all these tasks, this component interacts with the database management component for data storage and retrieval.
- **Reputation computation engine:** The reputation computation engine is the core of the Reputation Service. The computation engine will communicate with the database management layer in order to retrieve the current reputation score, the credibility of the

Co-funded by the Horizon 2020 Framework Programme of the European Union

evaluating user etc. and store their updated values. For the calculation of the updated credibility and reputation value, the reputation computation engine will handle the retrieval of monitoring and SLA data from the respective APIs. The computation engine will receive user evaluations from MySlice and jFed tool and will supply them with the updated reputation values to be presented to the users through the GUI.

- **Database management – Reputation service repository:** These two components consist of the Ruby on Rails models, the Active Record O/RM and a PostgreSQL Database. The models will be used for data initialization and validation and other database operations. All data will be stored and retrieved form the PostgreSQL database.

Components	Technologies
Testbed management – Testbed controller	Rails 5.1.4 Ruby 2.4.2
Database management	Active Record 5.1.4 Postgre PostgreSQL 10.0

Table 2: Reputation service technical requirements

1.5 MYSLICE

1.5.1 MySlice components & Software/Tools

The new architecture is composed of 5 layers with a clear separation of concerns: Web frontend, APIs (REST/WS), Database, Services with workers and Library (XML-RPC).

Co-funded by the Horizon 2020
Framework Programme of the European Union

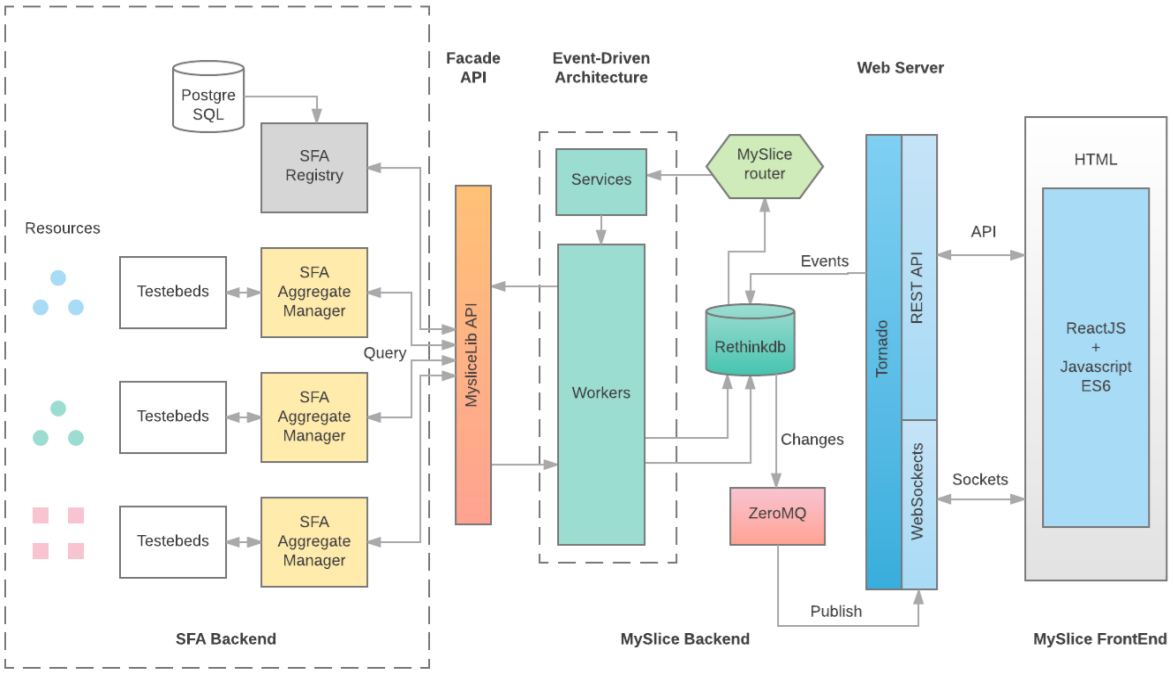

Figure 2: MySlice v2 Architecture

The frontend has been redesigned using the ReactJS framework. The benefit of using such a framework is to create generic components that can be re-used in different views depending on the properties passed to the components. Moreover, the management of a store that maintains a state of a component or a view is very well suited for an event-oriented application.

We have clearly defined the REST and WebSocket APIs used by the React components and third-party software. The web components are able to get or post data through the REST API and can be notified of a change through the WebSocket, providing a very interactive frontend.

Some interactions of the user with the frontend generate events that are stored in a document oriented database. The MySlice router is then responsible to place these events in the relevant queue depending on their type. Each type of event is asynchronously processed by a service. The services call workers that can be multithreaded to scale up the capabilities of the system. The workers are responsible of the interactions with the distributed testbeds through the AM API (XML-RPC) and with the SFA Registry, which is the root authority of the federation providing the credentials to access the testbeds.

As presented in the architecture, MySlice v2 features a database that stores the information about the objects of the federation as JSON documents. Synchronization processes periodically refresh the data of this caching system. Depending on the object, the periodicity ranges from once a day for the list of authorities at the SFA Registry to once every 5 minutes for the list of leases at the AMs that support scheduled reservations of resources.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Components	Technologies
Myslice, Services, WS, views REST API :	Python 3.5 RethinkDB 2.3.5 Tornado 4.4.0 JavaScript ES6 React 15.2.1 Alt.js 0.18.4
MysliceLib	Python 3.5
SFA Registry	Python 2.7

Table 3: MySlice v2 technical requirements

1.6 FRONTEND

The frontend tools (GUIs) of Fed4FIRE were MySlice and jFed. The jFed experimenter GUI and command line interface (CLI) is a Java based client that allows experimenters to reserve and instantiate resources, create topologies of the experiment. Furthermore, MySlice is used by the federation's web portal for creating slices and discovering and reserving resources.

The activities of Task 3.1 include the integration of the SLA and Reputation Service with the jFed tool and MySlice. This will be achieved by developing an appropriate plugin for the federation portal (MySlice) and an extension of jFed GUI and CLI.

1.6.1 SLA Frontend

It already has been introduced in section 1.3.1 that the SLA frontend component is called *SLA Dashboard*. It offers web-based GUI for two main functionalities for testbed providers. On one hand the functionality of creating templates that contain the basic definition of the offering. On the other hand, they will be able to check the status of the agreements that are being executed using their resources.

Testbed owners will need a certificate that identifies them within the system that grants the access to the web site. Once they are in the site, they might create the conditions of their offerings from the testbed.

Creation of an offering

An offering or SLA template includes information such as the guarantee terms. They are the conditions the platform owner will fulfil if somebody wants to use the offering. When an SLA agreement is created based on the SLA template, the guarantee terms remain the same; since

Co-funded by the Horizon 2020 Framework Programme of the European Union

the negotiation is one-shot the experiment will accept or reject the conditions included by the testbed owner (the provider).

The *SLA Dashboard* must include the option to create a template and its guarantee terms. Each testbed owner has to use metrics that can be measurable from the federation when creating the guarantee term and set the threshold that has to be guaranteed. For example, a provider would be able to specify that the minimal availability will be 95% as guarantee term for a SLA template. The testbed owner must ensure that he is able to measure the availability rate and that it will be included in the monitoring data. The threshold must be expressed by the same unit as the monitoring data. The representation of the guarantee in the SLA template would be: availability_rate > 0.95.

Status of the agreements

Another key functionality of the dashboard will be the presentation of the status of both active and past agreements. The testbed owner should retrieve a list of all agreements that have been created in his testbed and check their execution. For each agreement, the owner should be able to retrieve the template of each agreement and if the agreement is active to be able to view which terms produce possible violations.

1.6.2 REPUTATION frontend

The Reputation Service aims at facilitating users to choose the most suitable testbed for their experiments. Apart from designing and implementing an efficient reputation algorithm for the testbed federation, the user's experience with the Reputation Service should be straightforward, easy. The frontend tools and GUIs (MySlice, jFed) play a key role in achieving this. Both tools should present in an easy way to compare, reputation score values for each testbed and their number of ratings in the resource selection dialogues. This will be implemented as a plugin for MySlice and as an extension to jFed. In order to increase the effectiveness of the Reputation Service, the users will be prompted to evaluate their conducted experiments at the end or the cancellation of their reservation. This could be achieved with a jFed prompt and via an e-mail alert for users not using jFed.

1.7 FUTURE WORK

Through the first cycle, the SLA components will be developed simultaneously with the development of the monitoring data APIs and the Reputation Service. This will be done in order to avoid any inconsistencies and incompatibilities between the different components.

The SLA component needs to interact with all the components both collecting and providing data. Hence, it has been decided to follow a bottom-up implementation, and start from simple scenarios, which has been introduced in Fed4FIRE project. First, we have to define the metrics that the monitoring system can provide, and the creation of simple agreements for these experiments for every testbed. Due to the decentralized architecture, the definition, integration and implementation of the components will be distinguished to the Testbed domain and the Federation domain components.

- Testbed domain. i) aligned with the testbed components: monitoring and the aggregated manager, ii) implementation of new SLA adaptors in order to use the new monitoring interfaces, iii) improvement of the installation guide and simplification of the tested deployment, iv) introduction of the aggregated monitoring data since the new monitoring version is also distributed v) identification and definition of new metrics together with the rest of components.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Federation domain: It has been split in two levels: the collector layer and the GUI interfaces.
    - Collector Layer. i) adaptation of the collector to expose the new functionalities, ii) integration with the experimenter's certificate iii) notification of the violations to the reputation component, iv) creation of the procedure to incorporate new testbeds
    - GUI Interfaces. i) integration with the jFed and Myslice components, ii) creation of the new SLA plugin in the new Myslice component, iii) improvement of the previous SLA Dashboard to incorporate the experimenters and the administrators of the testbeds, iv) authentication based on the experimenter certificate.

During the first cycle of Fed4FIRE+, the new reputation algorithm will be developed, and a prototype of the new reputation service will be implemented in NETMODE and integrated with NITOS testbeds. This requires the following steps.

- Designing and testing the new reputation algorithm.
- Development of the Reputation Service web-app and API.
- Development and deployment of the monitoring data APIs in NETMODE and NITOS testbeds.
- Installation of the Testbed domain SLA components in NETMODE and NITOS testbeds.
- The interconnection of the Reputation Service and the SLA and Monitoring Data APIs.

At the second cycle, the reputation service will be incrementally implemented in other Fed4FIRE+ testbeds, while at the last cycle the final version of the reputation service will be integrated with jFed and MySlice tools.

1.8 REFERENCES

[1] Kapoukakis, A., Kafetzoglou, S., Androulidakis, G., Papagianni, C. and Papavassiliou, S., 2014, December. Reputation-Based Trust in federated testbeds utilizing user experience. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on* (pp. 56-60).

Co-funded by the Horizon 2020
Framework Programme of the European Union

2 YOUREPM – EXPERIMENT ORCHESTRATION

2.1 FUNCTIONAL REQUIREMENTS

Nowadays, the collaboration between different domains is necessary to achieve successful results and provide added value. Products, solutions and experiments are more transversal than some time ago, and the architectures foster the use of decouple components and services, exposing interfaces to be interconnected. Distributed topologies have been introduced increasingly by the using of cloud solution allowing to distribute and use solution as software as a service, without having a deep knowledge about how to install or maintain, only interested in consuming their functionalities.

Therefore, orchestration of services is more and more necessary for several reasons such as heterogeneous solutions, not isolated realms, more complex solutions, lack of knowledge in all the ambits, flexible and modular solutions… Fed4FIRE identifies these trends also for experiments, fostering the concept of EaaS (Experiment as a Services). Hence, the introduction of the orchestration tools, such as YourEPM, will allow orchestrating experiments, providing more complex experiments through transversal domains. YourEPM is based on the standard BPMN, which allow combining automatic and manual task in a visual way. Due to this, this orchestration layer over all the Fed4FIRE+ architecture will provide added value and improve the existing ecosystem.

The aim of adding orchestration of experiments is to enable more complex solution, combining different testbeds in the same experiment, plus third party services such as ticketing system, file storage… The tool has to manage the complete lifecycle of the orchestration from the design to the execution phase.

In order to achieve this, the orchestrator will need to cover the following functional requirements:

Id:	YourEPM-01	**Type:**	FUNC	**Priority:**	High
Title:	Support Multi-Tenant				
Description:	The solution has to manage multiples tenants using the same instance. This allows introducing concepts such as reusability and EaaS-based solution for Fed4FIRE+. Communities, companies or universities can be use the same tool to orchestrate their workflows.				
Additional Information:					

Id:	YourEPM-02	**Type:**	FUNC	**Priority:**	High

Co-funded by the Horizon 2020 Framework Programme of the European Union

Title:	Design of complex experiments
Description:	The component has to allow the definition of complex experiments, where several steps interact with automatic actions (different experiments) and manual task (supervision of results, validation of supervisors...). The design will use a graphical interface, which simplify partially the complexity of the interactions and automatic actions.
Additional Information:	This functionality is cover only in the design phase and it should be completely separated from the execution phase (operational).

Id:	YourEPM-03	**Type:**	FUNC	**Priority:**	High
Title:	Assistant in the selection of the exposed services.				
Description:	The component has to be integrated with the service directory in order to facilitate the identification of the services, which should be integrated, and creation of the complex experiments.				
Additional Information:	There is a component called Service Directory, where the testbed can register the different services.				

Id:	YourEPM-04	**Type:**	FUNC	**Priority:**	High
Title:	Deployment of complex experiments (workflows)				
Description:	The component will allow deploying the workflow, which defines the complex experiment.				
Additional Information:	The component should allow to deploy by tenant and only the owner can see the deployed workflow.				

Id:	YourEPM-05	**Type:**	FUNC	**Priority:**	High

Co-funded by the Horizon 2020 Framework Programme of the European Union

Title:	Workflow Instance Management
Description:	The component allows managing and following the status of workflow instances, according to the workflow description in complex experiment.
Additional Information:	The instance only can be managed by the owners of the deployed workflows.

Id:	YourEPM-06	**Type:**	FUNC	**Priority:**	High
Title:	Manage the workflow execution.				
Description:	The component has to manage all the necessaries actions, when the workflow instances are running, such as service task (invoke experiments or third party services) and user tasks (interact with the experimenters).				
Additional Information:	The actions of the instances only can be managed by the owners of the deployed workflows.				

Id:	YourEPM-07	**Type:**	FUNC	**Priority:**	High
Title:	Manage the workflow engine environment.				
Description:	The component has to manage and follow up the complete workflow environment such as the deployed workflows, the executed instances, the different groups or departments, the management of tokens and the different tenants.				
Additional Information:	The complete lifecycle will be managed by the administrator of the environment.				

Id:	YourEPM-08	**Type:**	FUNC	**Priority:**	High
Title:	Integration with the authentication and authorization of Fed4FIRE+				

Co-funded by the Horizon 2020 Framework Programme of the European Union

Description:	The component will delegate the authentication and authorization to Fed4FIRE security solution, following user certificates.
Additional Information:	

Id:	YourEPM-09	**Type:**	FUNC	**Priority:**	High
Title:	Management of experimenters and their departments by tenant				
Description:	Every tenant will manage the workflow usage and they will assign the internal departments for the different experiments (workflows). Besides, the component will synchronize automatically the users, roles and tenants of user accounts.				
Additional Information:	The tenant only can manage his own experimenters and departments.				

Id:	YourEPM-10	**Type:**	FUNC	**Priority:**	High
Title:	Management of departments by tenant				
Description:	Every tenant will manage the workflow usage and they will assign the internal departments for the different experiments (workflows). Besides, the component will synchronize automatically the users, roles and tenants of user accounts.				
Additional Information:	The tenant only can manage his own experimenters and departments				

Id:	YourEPM-11	**Type:**	FUNC	**Priority:**	High
Title:	Internal management				

Co-funded by the Horizon 2020 Framework Programme of the European Union

Description:	The component has to manage internal users to cover administration functionalities. This allows to manage it without depending on the external authentication, for example, if a problem appears with the Fed4FIRE security system, it would impede to connect and analyse the problematic situation.
Additional Information:	

Table 4: YourEPM functionalities

2.2 EXPERIMENT SERVICE ORCHESTRATION

In Fed4Fire+ we want to include an experiment service orchestration solution to allow high level application service orchestration within the federation, i.e. help experimenters to design and execute in an easy manner cross-testbed processes, even execute services from 3rd party providers. The most important component from the orchestration solution is called 'YourEPM' (Your Experiment Process Model).

YourEPM is a tool based on Activiti BPMN 2.0 software, which is java-based open-source software designed for Business Process Management and follows open standards such as BPMN (Business Process Model and Notation). These characteristics will be inherited by YourEPM. Activiti is not used as a whole in YourEPM, it's just the subset which best matches the needs from Fed4Fire+ that is being used. Some of the adaptations we will need to do for Fed4Fire+ is: integrate with the security solution provided within the project, be multitenant and change the web-interface to support it, enable the connection with the services provided by the testbeds from the project.

The other component from the Experiment Service Orchestrations solution, but with less functionalities, is the Service Directory. It is a simple component where testbed providers can publish the information about their public services. YourEPM is able to retrieve the information from the Service Directory, simplifying the definition of the tasks (automatic and manual) and the own experiment.

Both YourEPM and the Service Directory will be independent components placed in a centralized manner in the federation domain. In case of YourEPM, the architecture is being designed in such a way, that two instances can co-exists in different environments. We will have the design environment and the production environment. This decision has been taken in order have independent execution of tests while designing from executions that can be done in production. While designing and testing an experiment an eternal loop can be introduced, hence we can block a whole instance of YourEPM. Therefore, we must allow stopping the design environment without influencing a process that is being executed for production.

The authentication is delegated to the Fed4FIRE+ security solution; hence the experimenters' certificate will be used to identify the user and his/her organization. Only one account in the ecosystem is recommended to be used by the users in order to improve the quality of the experience.

Co-funded by the Horizon 2020 Framework Programme of the European Union

On the other side the orchestration of services requires unattended calls to execute the automatic task in the definition of the workflows. This kind of call will be executed by the own workflow engine without any human interaction, so it is necessary to allow the engine to execute the call on behalf of an user or company. There is a wide range of possible solution that have been directly decided by who expose the service, then the orchestration tool have to be as versatile as possible. In Fed4Fire, we have introduced the Speaks-for component that allows the experimenters to use the services provided by the testbeds in a unified manner, acting as a wrapper for the service. Nevertheless, we could also integrate with other third party services outside Fed4FIRE ecosystem and they are using other kind of solution like OAuth2 tokens, basic authentication, bearer tokens… Therefore, the integration will not be able to cover all the possible solutions, but it will be aligned with Fed4FIRE+ (based on Speaks-for solution) and the necessary third party services that the project wants to integrate.

The orchestration service will be integrated with services that have been exposed RestFull interfaces. Some testbed owners might implement a RestFull interfaces for their services, but some of them not. For those who do not provide a RestFull interface we will use the jFed component that will act as a wrapper for the service translating from the API REST to SFA.

The following figure depicted the architecture diagram.

Co-funded by the Horizon 2020
Framework Programme of the European Union

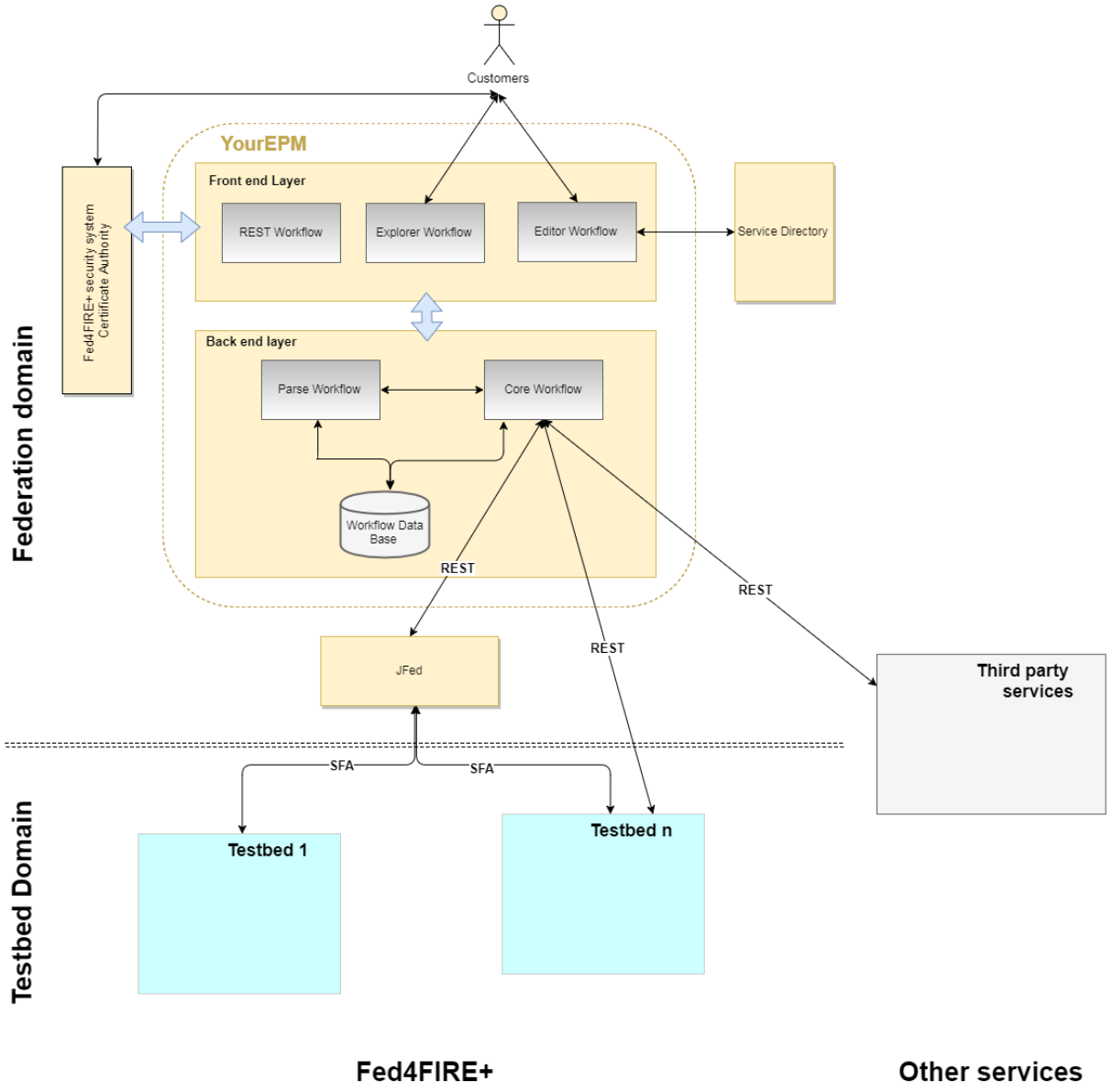

Figure 3: YourEPM Architecture

2.2.1 Service Orchestration Components

As it is already commented in the previous section, two different components will provide the Experiment Service Orchestration within the federation. These components are the Service Directory and YourEPM:

It has been already mentioned that the **Service Directory** is a quite small component that will record the service description and the URL from a service provided by testbed.

The **YourEPM** component covers more functionalities needed in fed4Fire+. It is design to cover different phases of the business process lifecycle: i) the design, where the workflows have to be created based on the complex experiments definition; ii) the deployment, where the defined workflows will be deployed iii) the execution, where processes are orchestrates.

Co-funded by the Horizon 2020 Framework Programme of the European Union

We have to keep in mind that YourEPM will be a centralized component and therefore it has to implement multitenancy to provide isolation among experimenters and its processes.

The YourEPM users will have assigned one or more roles which will enable them to perform different tasks:

- *Super-user* will be able to create the organizations which are allowed to use the YourEPM solution.
- *Administrators* associated to an organization will be able to request the tokens to access to the services from a platform or to a 3rd party service and they can manage the experimenters of their organization and the different departments involved. They will be also able to check which tasks have not been assigned to anybody like the experimenters.
- *Experiment designers* will be able to design and test new orchestrated experiments. They will be able to assign the tokens acquired by the administrator to be used to authenticate the service execution; however, they only will work with the design environment in order to provide the necessary testing before to release the final version of *the composed experiment.*
- Experimenters will be able to execute a deployed workflow (experiment). They can claim the manual task assigned to their department for their organization. The usage of token already associated to their organization to request the execution rights will be transparent for the experimenters. They only need to focus on execution of the experiment and the results.

Only the super-user will be pre-existing user in YourEPM database. Any other users from any organization will be created on-the-fly when it logs-in for the first time. The user will present his Fed4Fire+ certificate in order to access to YourEPM GUI. This certificate will include information such as the organization that the user belongs to and it might include information of the user's role (at least administrator or simple user). YourEPM will extract the information from the certificate and store it locally.

To access to any service from any testbed provider a speaks-for credential has to be presented. We already have commented that the administrator will be in charge of requesting the speaks-for to the testbed in order to authenticate the organization. If needed, they will be able to add other authentication mechanisms from 3rd party software like Dropbox and the generation of OAuth2 tokens. For example, a designer would be able to request that the results from the execution of an experiment are to be recorded in Dropbox.

The Activiti BPMN 2.0 does not include RestFull invocations for the automatic tasks (services tasks), this is and enhancement that will be done and will be one of the functionalities that will be supported by YourEPM. A proxy will be implemented that will be able to make RestFull calls to any component with such interface. During the design of an experiment it has to be specified that the RestFull proxy has to be used for the service invocation. From the Service Directory, we will retrieve the URL to be used by the service. Additional parameter might be specified to be included in the RestFull call. During the experiment execution, the RestFull proxy will include data like the token and execute the actual call to the service.

The components are decoupling between presentation layer, which is directly related to the exposed interfaces (Rest API and graphical user interfaces) and the backend, which is responsible for providing all the functionalities and the respective data persistence. Hence, the interface layer is responsible for providing the look and feel and how to expose its interfaces and it is also responsible for managing the necessary calls to the backend layer, which will execute the corresponding actions needed.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Front End layer is responsible for exposing the interfaces with the experimenters:

- *REST Workflow* module is responsible for exposing all the functionalities through a REST API, allowing other components to interact with the engine pro-grammatically, without human interaction.
- *Explorer Workflow* module is responsible for exposing a graphical user interface in order to interact with the experimenters. Hence, through this exposed dashboard, the different actors can manage the workflows, allowing interacting with the platform in an easy way, increasing the quality of experience (QoE). The authentication will be delegate to the experimenters' certificates managed by the Fed4FIRE+ security system.
- *Editor Workflow* module is responsible for the editing of the workflows at the design phase. A graphical user interface is used for editing, modifying and generating the workflows, while the involved actors are the technical designer, who can be contracted by the brokers to take care of the workflow design task. It interacts directly with the Service Directory to integrate the exposed services.

Back End Layer is responsible for managing the business logic:

- Core Workflow Engine module is responsible for managing all the functionalities of the Workflow Engine. Hence, the complexity of the business logic of all these functionalities is delegated to this module and an interface is exposed to interact with them. It is also responsible to interact with the data layer and persist the workflows, its instances and the rest of the entities.
- Workflow Parser module is responsible for managing the functionalities related to workflow parsing, such as automatic generation of the service task tags to invoke WS and RestFull services.
- Workflow Data Base module is responsible for persisting all the data in order to support all the functionalities. The model definition is based on an entity-relationship schema to represent all the entities, including tenants, roles, workflows, instances and jobs.

YourEPM technical requirements are the same as the ones from Activiti BPMN 2.0 which is a java based component. Activiti can work with different databases, but in Fed4Fire+ we will only implement the required functionalities for MySQL database. In addition, we're going to use the web-based solution that has to be installed in an application server like tomcat.

The **Service Directory** is a java based application that runs behind and application server. The data is also stored in a MySQL database; therefore the requirements are the same as YourEPM component.

Components	Technologies
Workflow Engine (YourEPM) Service Directory	Oracle JDK >=1.7 Database: MySQL>=5.0 Application server: Apache Tomcat: >= 6.0

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 5: YourEPM technical requirements

Co-funded by the Horizon 2020
Framework Programme of the European Union

3 AUTHENTICATION PROXY SERVICE

3.1 REQUIREMENTS

When building services (such as YourEPM) on top of the federation tools and testbeds, we want the following:

- Users should be able to use their Fed4FIRE certificate (instead of needing another authentication method/credentials)
- In Fed4FIRE, users are part of projects (sub-authorities), possible with other users. Services should have access to this information.

In a more technical way, we want:

- To use Fed4FIRE credentials to authenticate and authorize users in a REST API
- Secure communications between client and server
- Create a reusable component to hide the complexity of certificate processing and SSL session handling from the backend service

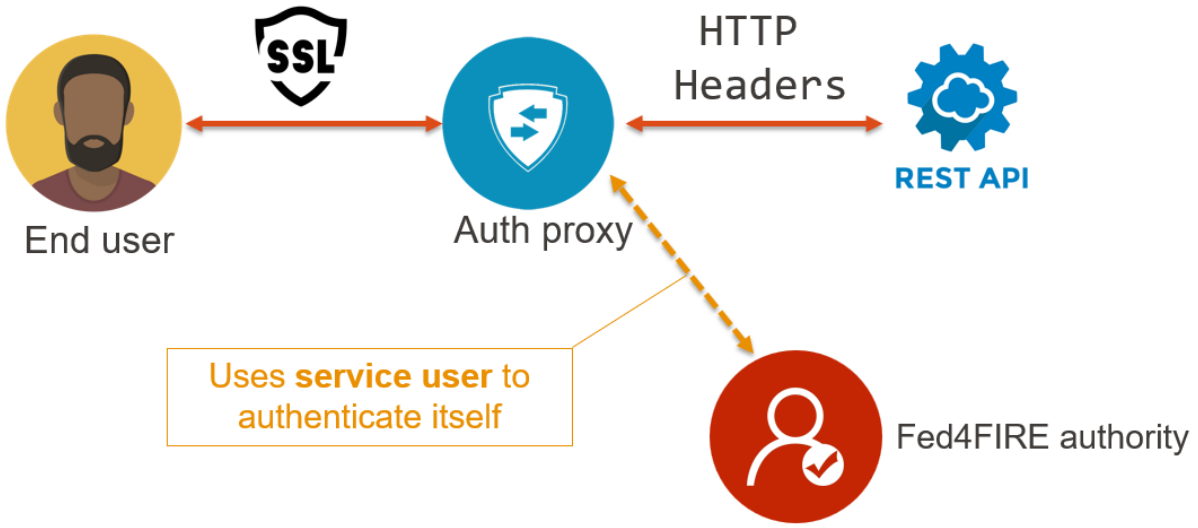

Figure 4: Architecture of authentication proxy service

Figure 4 shows the different parts in the architecture of such an authentication proxy service:

- Authentication proxy:
  - Handles SSL connection termination
  - Authentication: verifies client certificate with Fed4FIRE authority
  - Authorization: fetches all user projects from Fed4FIRE authority
  - Forwards this information to the backend service by injecting HTTP Headers into the request
- REST API: this is the REST API of a 3rd party service

Co-funded by the Horizon 2020 Framework Programme of the European Union

3.2 EXAMPLES OF INJECTED HTTP HEADERS

- **Fed4Fire-Authenticated**: True
- **Fed4fire-Authenticated-User-Urn**: urn:publicid:IDN+wall2.ilabt.iminds.be+user+twalcari
- **Fed4fire-Authenticated-User-Projects**: urn:publicid:IDN+wall2.ilabt.iminds.be+project+fgre, urn:publicid:IDN+wall2.ilabt.iminds.be+project+fec1, urn:publicid:IDN+wall2.ilabt.iminds.be+project+twalcari-test

3.3 ADVANTAGES AND DISADVANTAGES

3.3.1 Advantages

- Reusable for all HTTP backends
- Removes complexity from backend code
- Secure SSL connection with client certificate

3.3.2 Disadvantage

- Access from proxy to backend must be sufficiently secured (bind to localhost, internal firewall) (this is not really a disadvantage, but rather a point of attention)
- Delay during first call while fetching projects from authority (this should be improved by making the authority more efficient). After the first call, this info can be cached by the proxy.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4 OTHER REQUIREMENTS & VISION ON REQUIREMENTS

When asking users for requirements, it appears they respond typically with very detailed demands (e.g. a button for doing X would be handy, we need more documentation on this). These things are handled by the developments of the Federator in WP2.

For the developments in WP3, where we want larger blocks of innovative developments, we need to tackle the requirements in another way, more specifically we need to extract them ourselves from looking at the experiments people are doing and where they struggle.

Based on this, we made the following analysis.

4.1 SCALING UP EXPERIMENTS

A lot of experiments are about scaling up the number of resources for testing prototypes. While scaling up is perfectly possible with the testbeds and tools available, it needs a fair amount of manual work to get his done.

As an example we take a plenary demo that was shown at GEC22 in Washington, where we combined multiple testbeds and scaled up to 1000 resources using multiple software defined exchanges for an international demo (Figure 5).

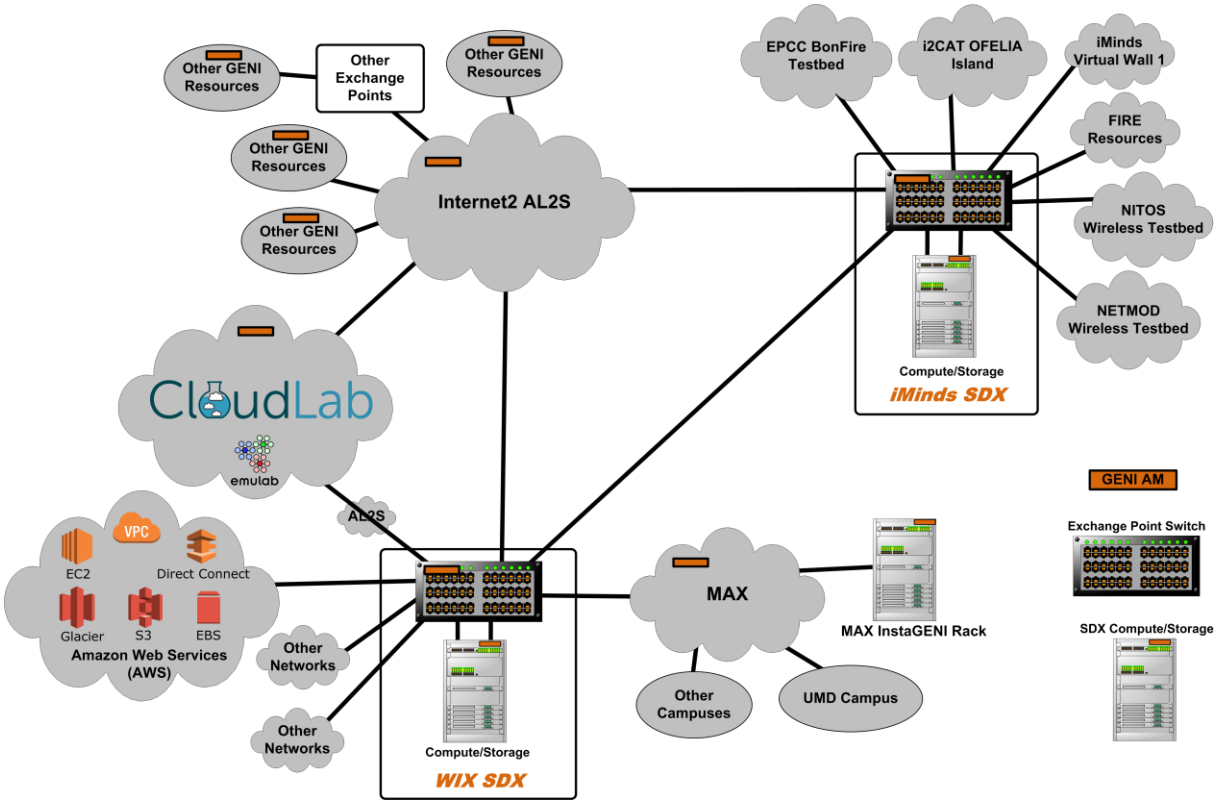

Figure 5: GEC22 scaling up demo

This demo was built in multiple steps:

Co-funded by the Horizon 2020 Framework Programme of the European Union

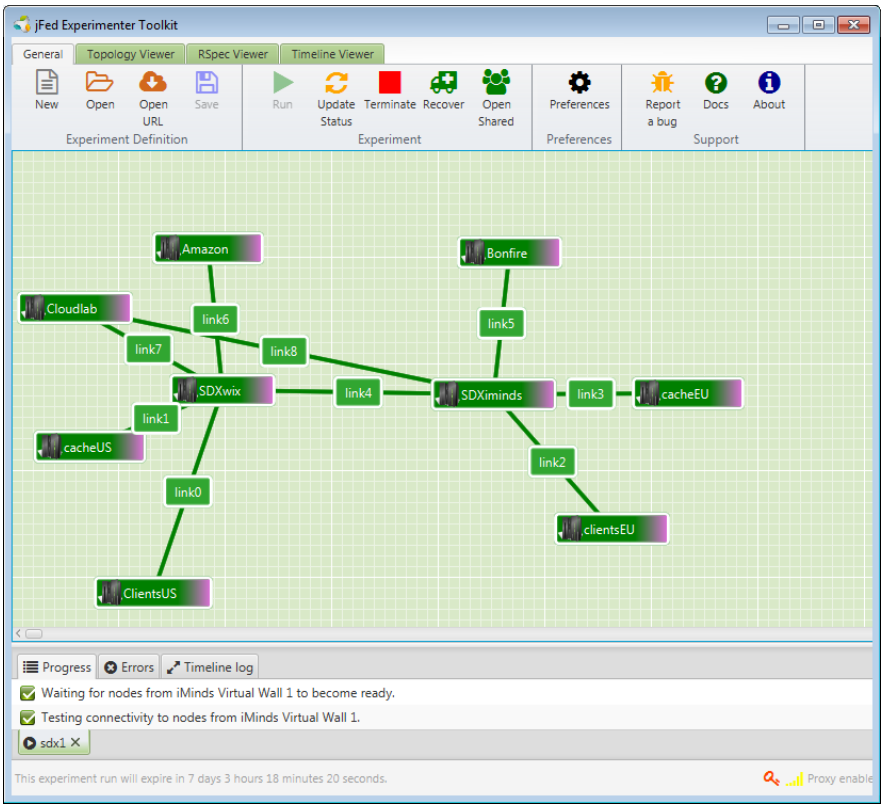

Figure 6: Build a (small scale) prototype on a single testbed to verify the functional aspects

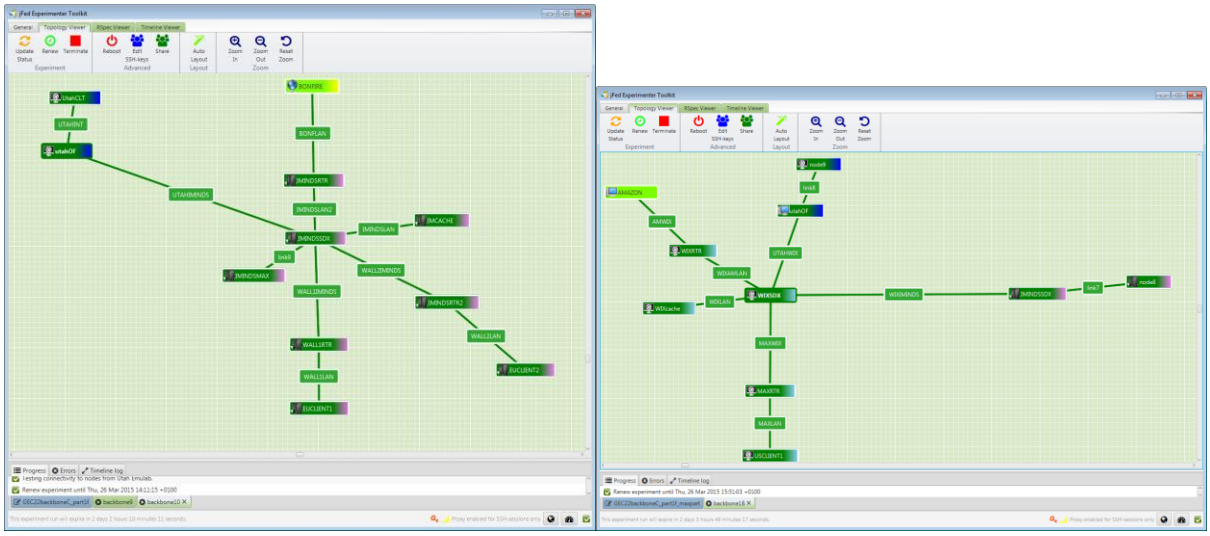

Figure 7: Build the backbone of the experiment on multiple testbeds

Co-funded by the Horizon 2020
Framework Programme of the European Union

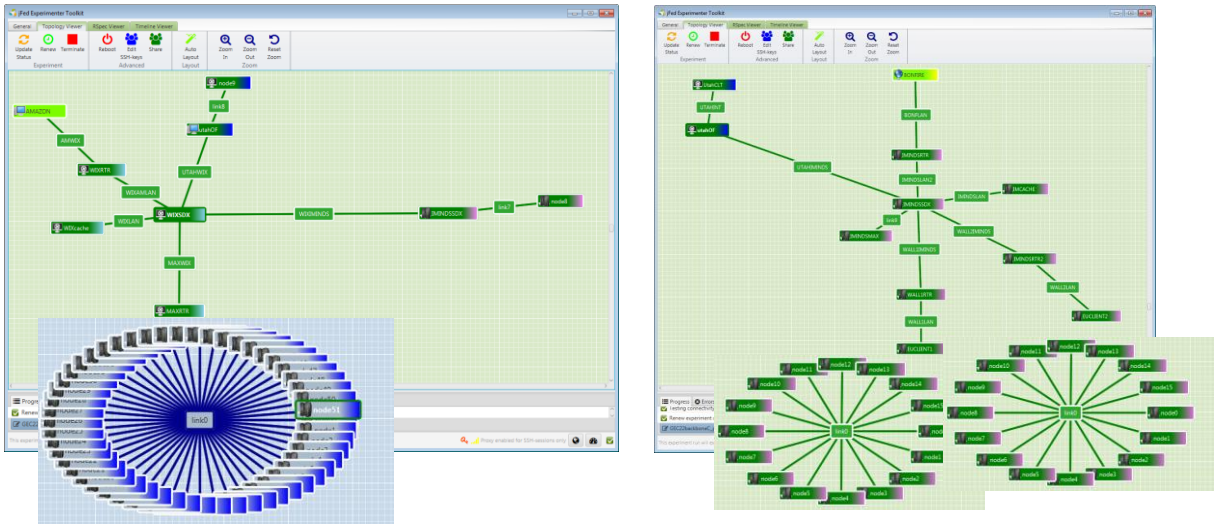

Figure 8: Scale up the number of resources

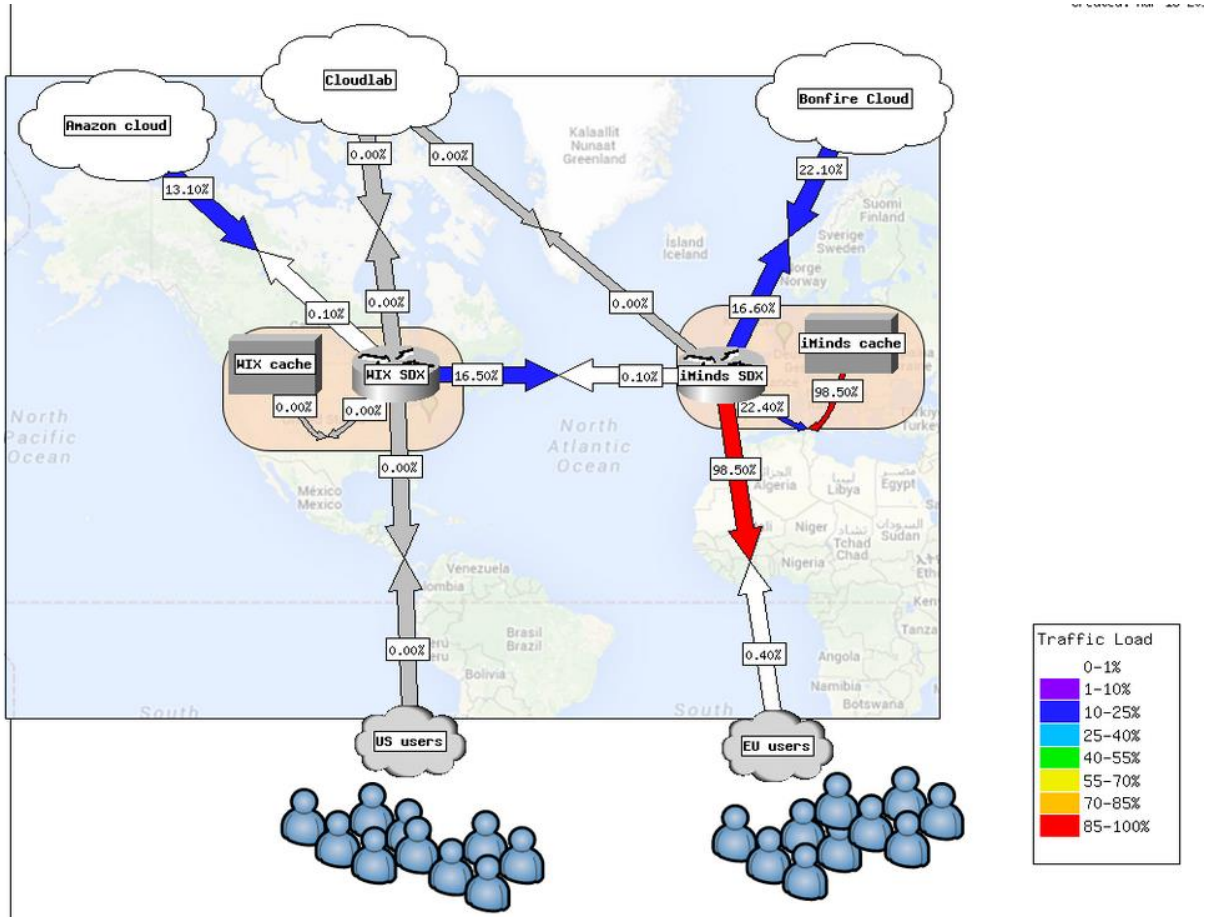

Figure 9: Do the actual experiment

We can conclude from this that it is possible to scale up to large amounts of resources, but it is not trivial for the average user, so we need something more simple.

Co-funded by the Horizon 2020
Framework Programme of the European Union

For this we will look into a more automatic way of scaling up, as shown in Figure 10. An Experiment Specification (ESpec) as an extension of the RSpec (Resource Specification) is used to set up all resources and infrastructure. And then in a 2nd step, 3rd party tools, such as kubernetes kubectl are used to start-stop containers to scale up. Adding a simple webinterface to this, makes it trivial for a user to scale to e.g. 10.000 containers.

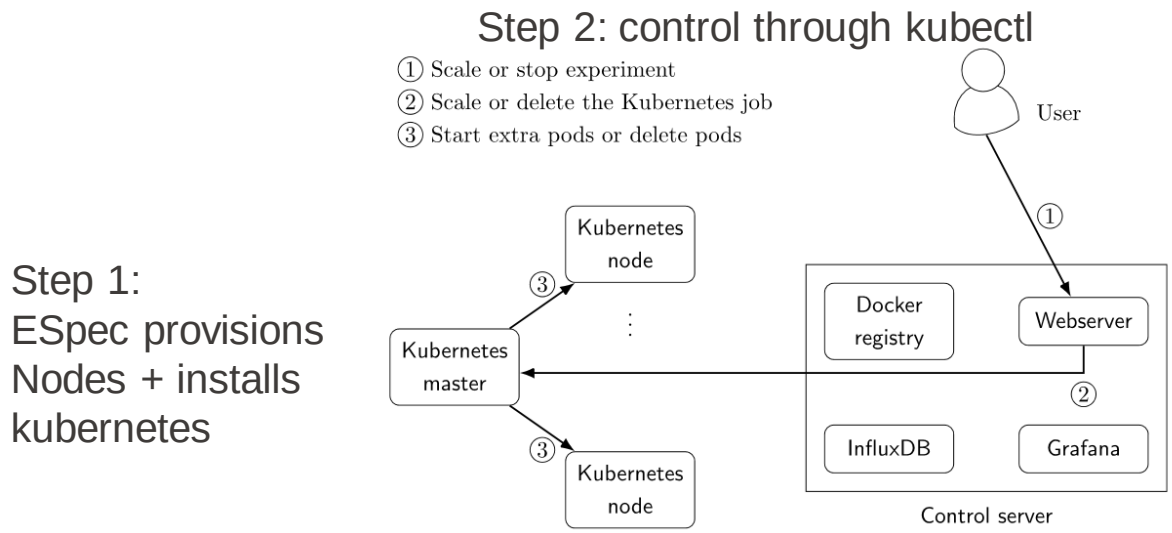

Figure 10: Scale up experiments through ESpec and kubernetes

A similar requirement, namely for 'large-scale recursive internet experiments' was identified in the ARCFire project and was solved by developing a tool called Rumba (https://arcfire.gitlab.io/rumba/). See also the publication on this: Sander Vrijders, Dimitri Staessens, Marco Capitani and Vincenzo Maffione, "Rumba: A python framework for automating large-scale Recursive Internet Experiments on GENI and FIRE+" .

Rumba is a python framework for Automating Large Scale Recursive internet experiments and helps in building easily large topologies, independent of the testbed used. Figure 11 shows the architecture of Rumba: multiple plugins are available for deploying the topology e.g. on local docker containers, testbeds (jFed plugin), or virtual machines. Also plugins are available for multiple new internet architecture frameworks (e.g. IRATI, Ouroboros, etc). Figure 12 and Figure 13 show examples of large topologies automatically deployed by Rumba.

Co-funded by the Horizon 2020
Framework Programme of the European Union

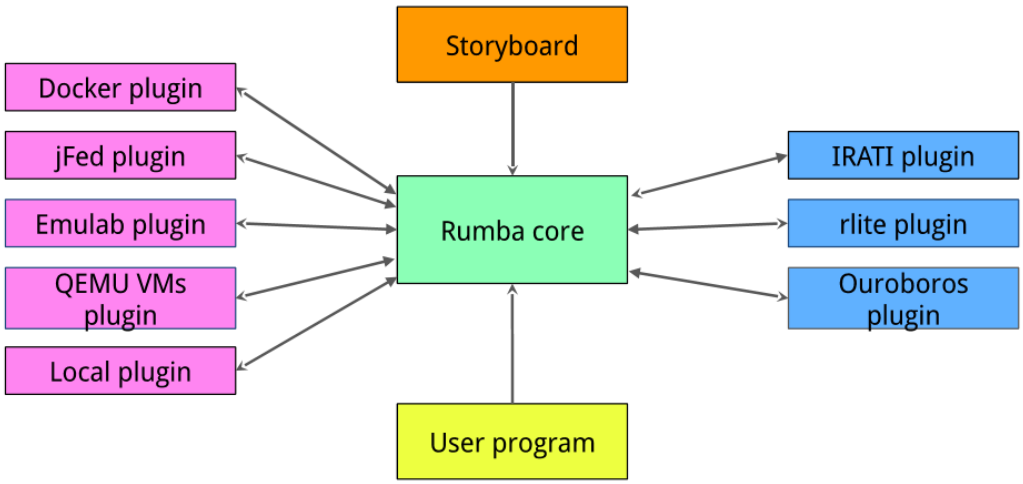

Figure 11: Rumba framework architecture

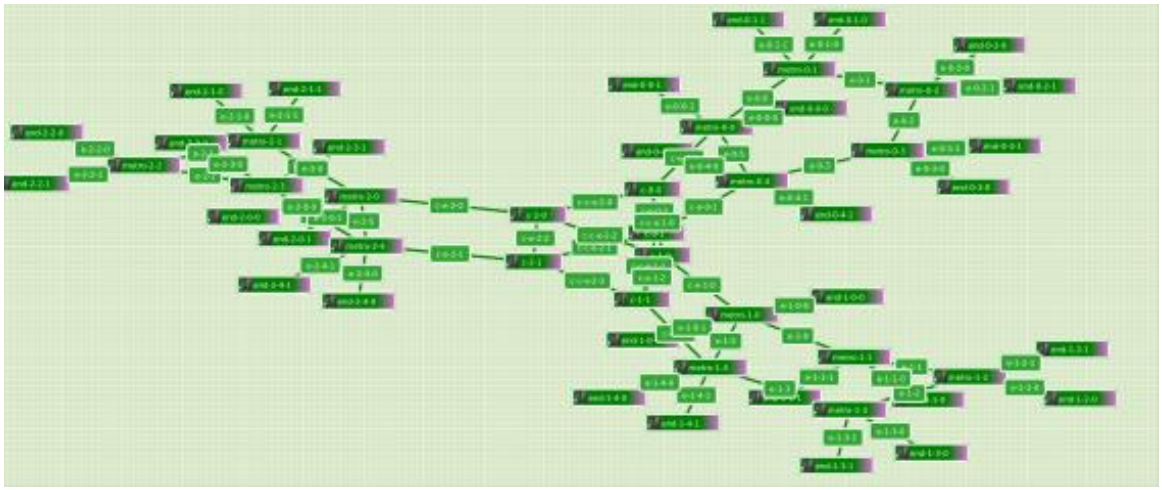

Figure 12: Large topology with multiple edge networks on the Virtual Wall testbed

Co-funded by the Horizon 2020
Framework Programme of the European Union

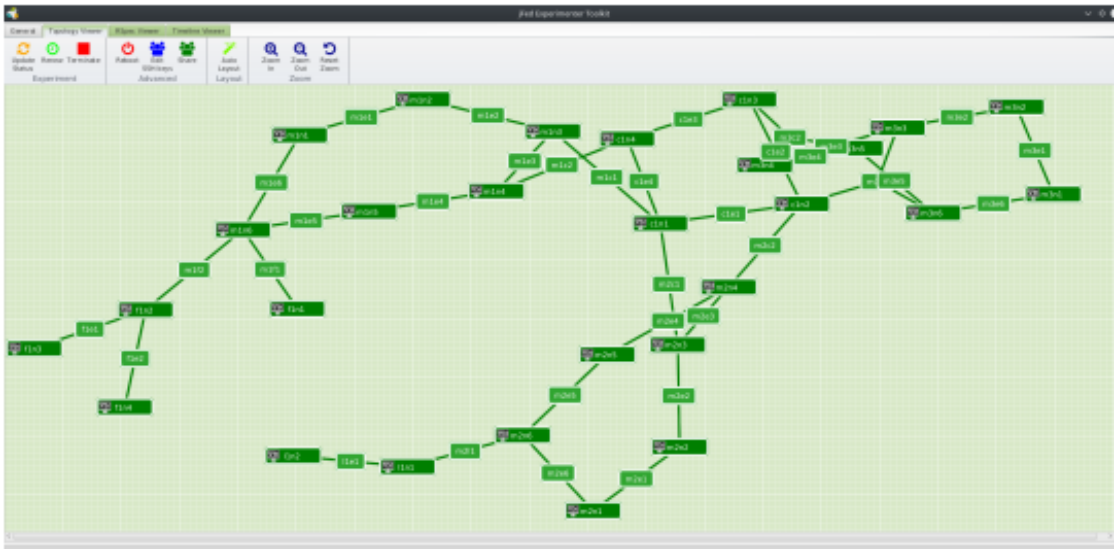

Figure 13: Large topology on exogeni

4.2 NFV/SDN EXPERIMENTATION

The Fed4FIRE testbeds and tools allow to do low-level NFV and SDN experimentation. E.g. an experimenter can use software tools as Click or Open vSwitch, or hardware SDN switches (Figure 14). Some of the experimenters create then their own tools (Figure 15) to make the NFV experimentation simpler.

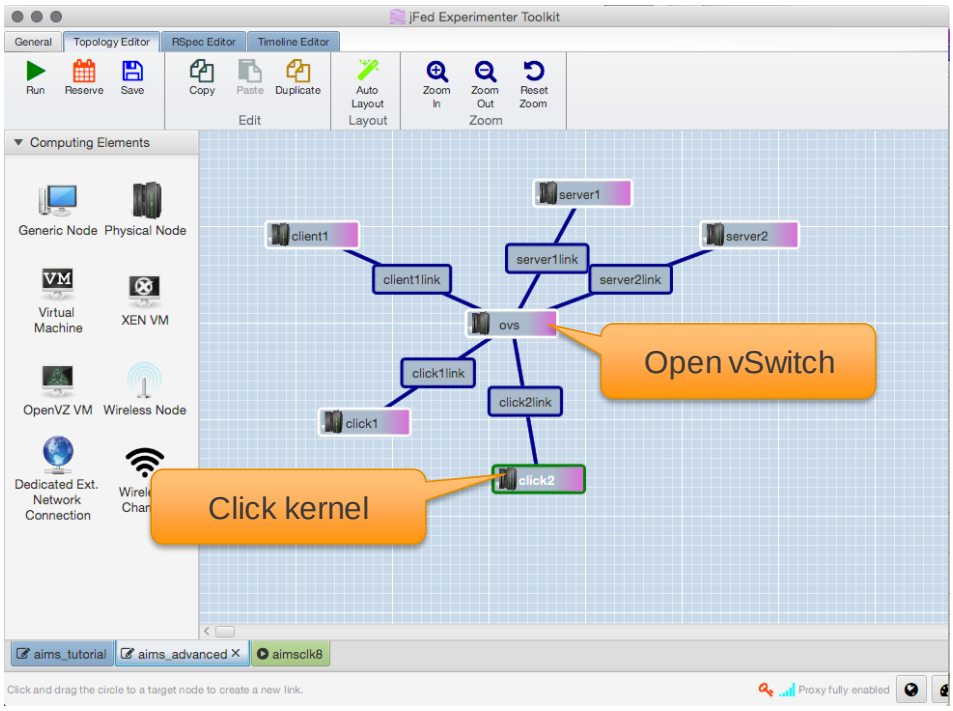

Figure 14: Manual experimentation with Click or Open vSwitch

Co-funded by the Horizon 2020 Framework Programme of the European Union

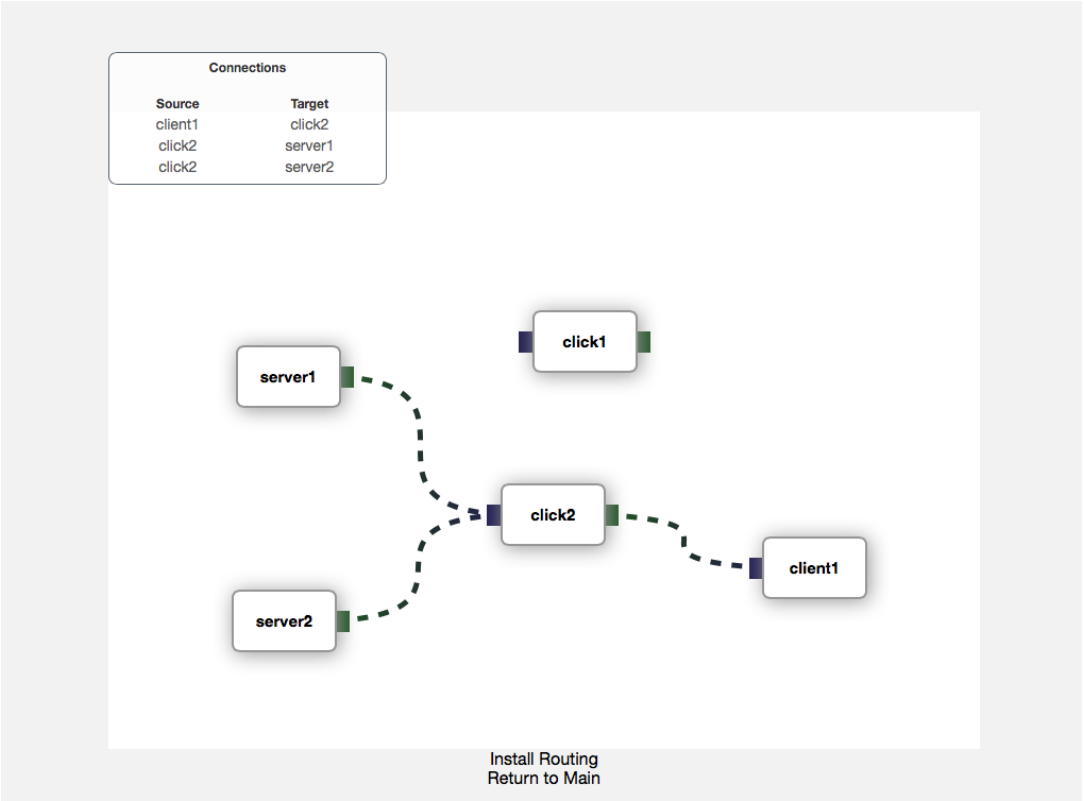

Figure 15: Experimenters creating their own frontend for easy NFV experimentation

In a similar way, other projects as e.g. Futebol (http://www.ict-futebol.org.br/) or Necos (http://www.h2020-necos.eu/) have added NFV/5G capabilities on top of Fed4FIRE testbeds and tools. Figure 16 shows how FUTEBOL uses jFed as a provisioning tool, and then uses other tools or frameworks such as Tosca or Copa to deploy containers or NFV functions.

Co-funded by the Horizon 2020
Framework Programme of the European Union

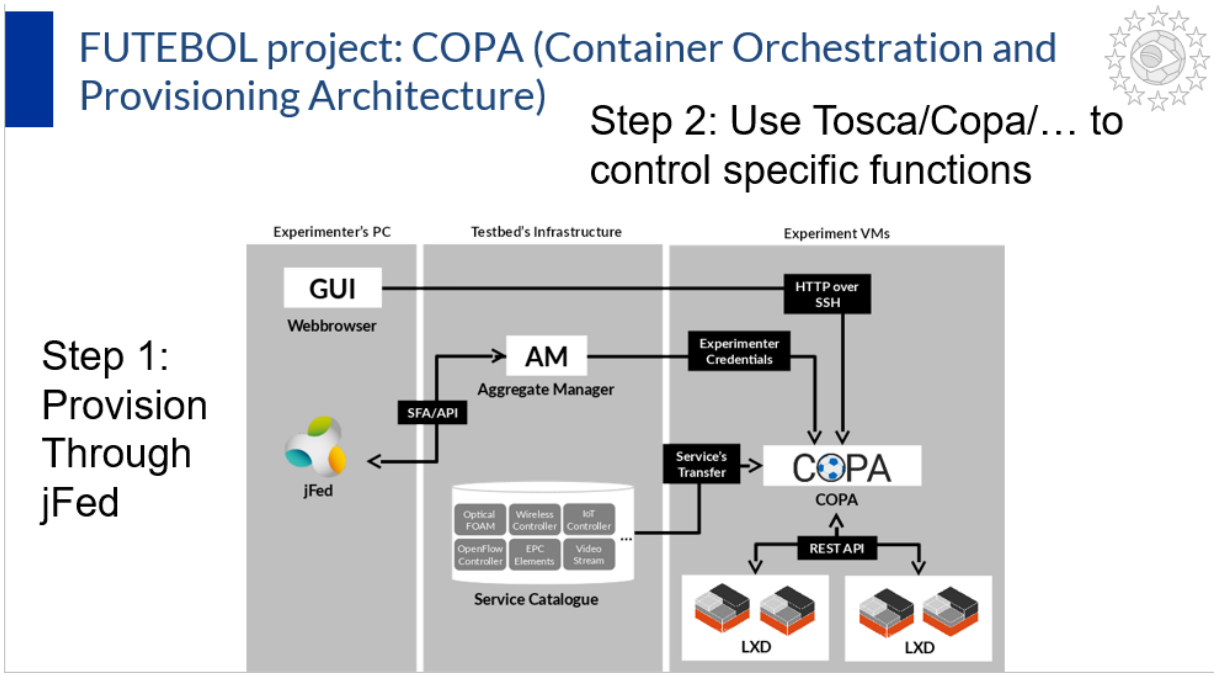

Figure 16: Example of Futebol project using the provisioning of Fed4FIRE with an added Tosca/Copa layer on top

4.3 AUTOMATE EXPERIMENTS

What we also learned from looking at experimenter and project needs, is the need for reproducibility and automation. Instead of rerunning again and again the same experiments (and forgetting steps each time), automating everything pays off in the long run for this kind of experiments. That's where the ESpec also comes in handy. This is e.g. needed for advanced software suite testing where testbeds are needed.

Figure 17 shows the example of the F-interop project (www.f-interop.eu). Remote users run test suites running on a central platform against IoT devices on their desks. If we now automate all this (deployment of central platform, test suites and using devices on testbeds), we have a continuous interop and conformance testing platform.

Co-funded by the Horizon 2020 Framework Programme of the European Union

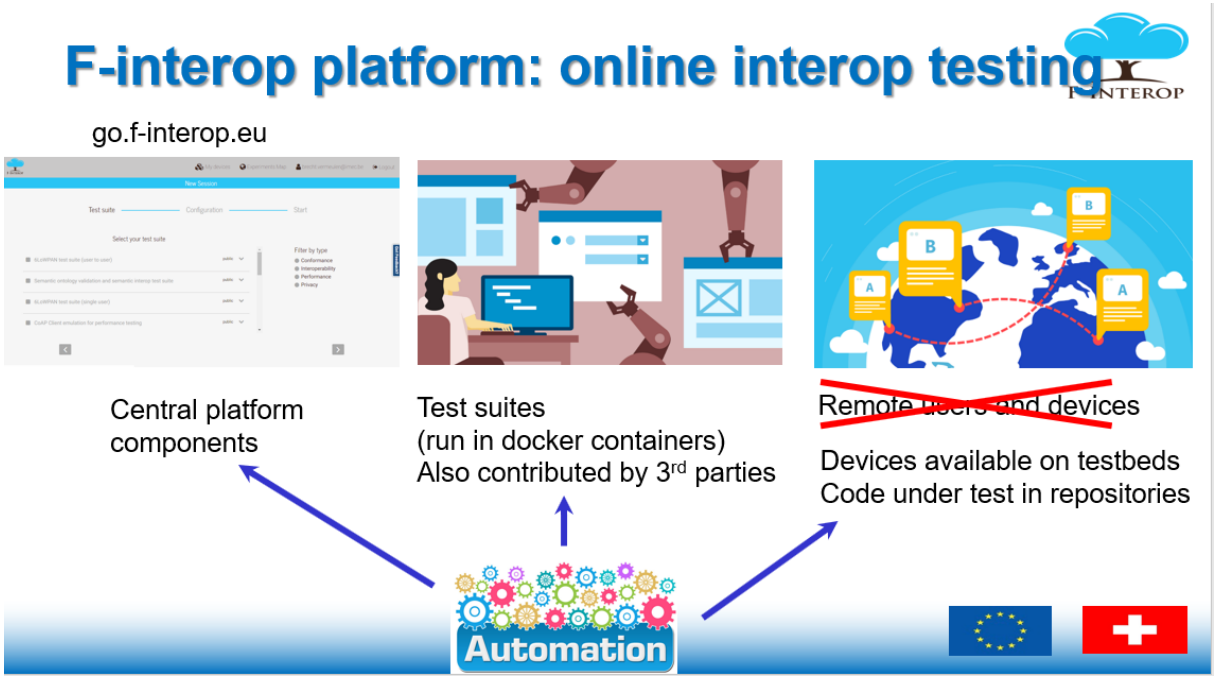

Figure 17: www.f-interop.eu remote interop and conformance testing

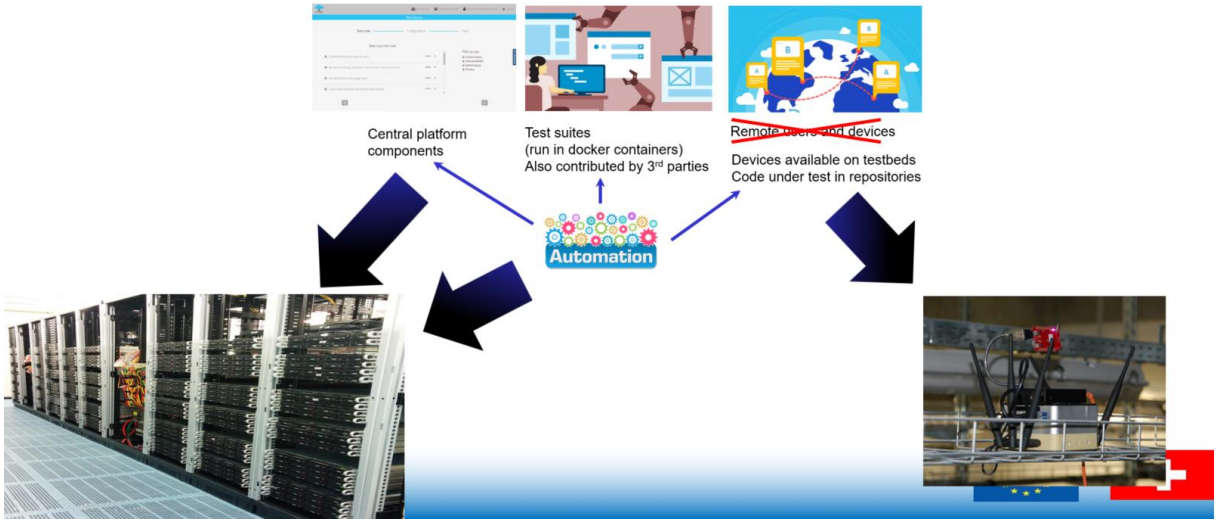

Figure 18: Running F-interop on testbeds

Co-funded by the Horizon 2020 Framework Programme of the European Union

5 CONCLUSIONS

In this deliverable we identified the (user) requirements for the developments in WP3. Detailed requirements for the SLA, Reputation and YourEPM modules were listed. An authentication proxy was identified as a module easing interaction with REST based services.

Besides those, we also looked from a bit further away, to identify needs of experimenters and we found out that Fed4FIRE testbeds do support all kinds of experimentation, but some experiments (e.g. scaling up, NFV/SDN, automation) can benefit from tools doing a lot of the work for the user.

In this regard, we see Fed4FIRE as a meta-testbed where others (e.g. other projects) can build tools on top. Key is then to bring these tools to production quality (with documentation, maturity, etc). D3.2 goes more into detail on some of these tools that have been implemented.

Co-funded by the Horizon 2020
Framework Programme of the European Union