Federation of Internet experimentation facilities: architecture and implementation

Tim Wauters, Brecht Vermeulen, Wim Vandenberghe, Piet Demeester, Steve Taylor, Loïc Baron, Mikhail Smirnov, Yahya Al-Hazmi, Alexander Willner, Mark Sawyer, et al.

Federation of Internet experimentation facilities: architecture and implementation

Tim Wauters[1], Brecht Vermeulen[1], Wim Vandenberghe[1], Piet Demeester[1], Steve Taylor[2], Loïc Baron[3], Mikhail Smirnov[4], Yahya Al-Hazmi[5], Alexander Willner[5], Mark Sawyer[6], David Margery[7], Thierry Rakotoarivelo[8],

[1]Ghent University – iMinds, Gaston Crommenlaan 8 Bus 201,
Gent, 9050, Belgium
[2]IT Innovation Centre, Southampton, UK
[3]Université Pierre et Marie CURIE (UPMC), Paris, France
[4]Fraunhofer FOKUS, Berlin, Germany
[5]Technische Universität Berlin (TUB), Berlin, Germany
[6]The University of Edinburgh (UEDIN), Edinburgh, UK
[7]Inria Rennes - Bretagne Atlantique (INRIA), France
[8]National ICT Australia Limited (NICTA), Eveleigh, Australia

Felicia Lobillo Vilela[9], Donatos Stavropoulos[10], Chrysa Papagianni[11], Frederic Francois[12], Carlos Bermudo[13], Anastasius Gavras[14], Dai Davies[15], Jorge Lanza[16], Sueng-Yong Park[17]

[9]Atos, Madrid, Spain
[10]University of Thessaly (UTH), Volos, Greece
[11]National Technical University of Athens (NTUA), Athens, Greece
[12]University of Bristol (UNIVBRIS), Bristol, UK
[13]i2CAT, Barcelona, Spain
[14]Eurescom GmbH, Heidelberg, Germany
[15]Dante Limited, Cambridge, UK
[16]Universidad de Cantabria (UC), Santander, Spain
[17]National Information Society Agency (NIA), Seoul, Korea

***Abstract* — Realistic experimentation facilities are indispensable to accelerate the design of novel Future Internet systems. As many of these ground-breaking new applications and services cover multiple innovation areas, the need for these solutions to be tested on cross-domain facilities with both novel infrastructure technologies and newly emerging service platforms is rising. The Fed4FIRE project therefore aims at federating otherwise isolated experimentation facilities in order to foster synergies between research communities. Currently the federation includes over 15 facilities from the Future Internet Research and Experiment (FIRE) initiative, covering wired, wireless and sensor networks, SDN and OpenFlow, cloud computing, smart city services, etc. This paper presents the architecture and implementation details of the federation, based on an extensive set of requirements coming from infrastructure owners, service providers and support communities.**

Keywords—Future Internet experimentation, federation, FIRE.

I. INTRODUCTION

As the Future Internet (FI) ecosystem is very diverse, the federation of experimentation facilities should support this variety as well. Multiple layers can be identified (see Fig. 1):

- Infrastructures: the core Internet industry consists of players aiming to provide a range of communications, processing and storage infrastructures.
- Services: the Services domain provides platforms offering utility functions for composing, controlling, managing, securing and billing for distributed service-based systems.
- Applications: applications combine and extend available services to deliver functionality to the users themselves.

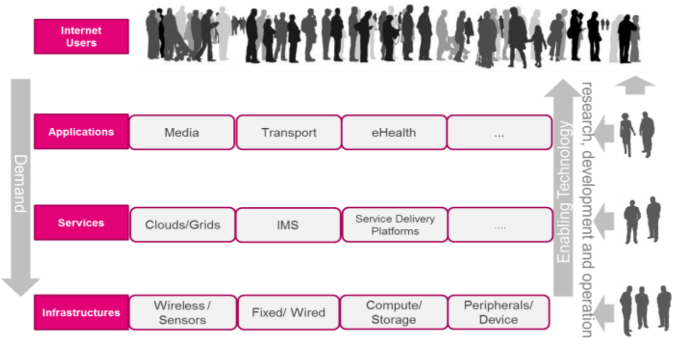

Fig. 1. Future Internet ecosystem

Within the Fed4FIRE federation, multiple facilities covering FI research areas are present[1], which are geographically distributed, also outside Europe. While some testbeds offer native infrastructure resources, others provide specific (IMS, EPC, data or cloud) services on top of their resources as well:

- Wired testbeds: Virtual Wall (iMinds), PlanetLab Europe (UPMC), Community-Lab (UPC Spain), optical access testbed (Stanford University, UC3M Spain).
- OpenFlow testbeds: UNIVBRIS OFELIA island, i2CAT OFELIA island, Koren testbed (NIA), NITOS (UTH).
- Cloud computing testbed: BonFIRE (UEDIN and Inria cloud sites, iMinds Virtual Wall testbed for emulated networks).
- Wireless testbeds: Norbit (NICTA), w-iLab.t (iMinds), NITOS (UTH), Netmode (NTUA), SmartSantander (UC), FUSECO (FOKUS), Community-Lab (UPC Spain), LTE performance lab (UMA Spain).

[1] http://www.fed4fire.eu/testbeds.html

TABLE I. EXPERIMENT LIFECYCLE

Function		Description
Resource discovery		Finding available resources across all testbeds, and acquiring the necessary information to match required specifications.
Resource specification		Specification of the resources required during the experiment, including compute, network, storage and software libraries.
Resource reservation		Allocation of heterogeneous resources spanning multiple testbeds based on a multiplicity of selection criteria (time, resource type, etc.).
Resource provisioning	Direct (API)	Instantiation of specific resources directly through the testbed API, responsibility of the experimenter to select individual resources.
	Orchestrated	Instantiation of resources through a functional component, which automatically chooses resources that best fit the experimenter’s requirements.
Experiment control		Control of the testbed resources and experimenter scripts during experiment execution. This could be predefined interactions and commands to be executed on resources (events at startup or during experiment workflow).
Monitoring	Facility monitoring	Instrumentation of resources to supervise the behavior of testbeds, allowing system administrators or first level support operators to verify that testbeds are performing correctly.
	Infrastructure monitoring	Instrumentation of resources by the testbed itself to collect data on the behavior and performance of services, technologies and protocols. This allows the experimenters to obtain monitoring information about the used resources that they could not collect themselves.
Measuring	Experiment measuring	Collection of experimental data generated by frameworks or services that the experimenter can deploy on its own.
Permanent storage		Storage of experiment related information beyond the experiment lifetime, such as experiment description, disk images and measurements.
Resource release		Release of experiment resources after deletion or expiration the experiment.

Although the cross-fertilization between research domains is an important driver for the federation, other major advantages can be identified as well. One example is the reduced cost for experimentation when resources are shared between the federation partners. Another one is the possibility to repeat the same experiment on different facilities, increasing the confidence in the results. A third example is the fact that due to federation, the widened resource offers of individual facilities can more easily attract industrial experimenters.

We present the current federation architecture that supports experimentation in the FI ecosystem, as designed in the second implementation cycle of the Fed4FIRE project. The path with the design choices to reach the first version of the architecture was presented earlier [1]. This architecture has been designed taking into account use cases and requirements from infrastructures, services and first level support communities, prioritizing those aspects considered critical. Service Level Agreements (SLAs) and sustainability aspects are taken into account as well. Furthermore, the architecture tries to take as much advantage as possible from existing tools and mechanisms in the facilities to be federated, which allows current experimenters to keep using their own tools with a broader range of resources, without imposing new ones.

The remainder of this paper is structured as follows: in Section 2 we describe the different aspects of the experiment lifecycle, and highlight the most relevant requirements in that regard. Based on this analysis, we present the current architectural view in Section 3. Finally, we conclude with a summary and outlook of the work.

II. EXPERIMENT LIFECYCLE REQUIREMENTS

As the design of the architecture is closely related to the lifecycle of an experiment, Table I provides an overview of the sequential actions the experimenter performs to help him or her to execute an experiment idea on a testbed and obtain the desired results.

On top of the requirements collected in the first cycle [1], novel requirements have been gathered from different communities related to infrastructures, services and first level support (FLS), as well as on SLA and sustainability aspects. As no operational data is available in the project so far (the first Open Call partners are starting their experiments in May 2014), experience on FLS has been gained from similar federated operational environments in the sector of backbone networks.

Additional sources considered in this process include specific research communities related to the experimentation facilities, proposals submitted to the first Open Call for experimenters and a set of novel experimentation use cases based on current commercial and research trends and specifically designed to push the envisaged federation tools to their limits in terms of scale and heterogeneity. In total, a set of 78 requirements was compiled in the second project cycle [2], out of which 61 are fulfilled by the architecture presented in the next section.

III. Federation Architecture

A. Introduction

The Fed4FIRE architecture should be able to cope with heterogeneous testbed software frameworks. Fed4FIRE's approach to achieving this is to focus on adopting the same (standardized) federation interfaces on top of the existing frameworks. This way tools can work with multiple testbeds, orchestration engines should only cope with a single type of interface, and user accounts can be shared over the testbeds. By grouping the requirements according to their functionality, different architectural components have been identified. The detailed architecture discussion is therefore split up into the following subsections:

- Resource discovery, reservation and provisioning
- Monitoring and measurement
- Experiment control
- SLA management and reputation services

During the design process, careful consideration was given to alignment with the work in GENI[2] in the US. As a result, the federation architecture does not only meet the requirements and other inputs defined above, but it is also interoperable with GENI.

B. Concepts of federation

In a federation, testbeds, services, experimenters and authorities have a common trust, although each individually can have its own policies. Federations can be small or short-lived, so they should be easy to set up or tear down. Examples of such federations are PlanetLab, Emulab and GENI, which in turn are also (partly) federated with Fed4FIRE. The specific actors in the federation are the testbeds (with resources and services), the experimenters (with proper credentials) and the federator (with central components). In Fed4FIRE, the federator only hosts components that make federation easier to use, e.g. directory services, but these components are not strictly needed for operations. Ease of use is not to be underestimated however. We envisage that most experimenters simply want their experiment to run with the minimum of effort and setup on their part. The federator can offer services such as a complete federated experiment setup and execution service for example, so experimenters can concentrate on what is important to them, rather than having to learn the intricacies of each testbed in their experiment. Therefore, we believe the federator offers a valuable service in making the federation easier to use.

The different horizontal layers in the architectural diagrams in the following section describe the levels of the testbed resources (physical and virtual), the testbed management (management of the local testbed resources), the federation services (federation services for operation or ease-of-use), the application services (testbed services, accessible for the experimenter and abstracting the underlying technical details) and the experimenter tools. The architecture figures contain multiple colors to categorize components: grey (a component which is mandatory), dotted (a component which is optional), green (an API which is also used outside of Fed4FIRE or which is standardized) and blue (a proprietary API for which there was an agreement in Fed4FIRE).

C. Resource discovery, specification, reservation and provisioning

Several aspects of this architecture originate from the Slice-based Federation Architecture (SFA) [2]: the Aggregate Manager API, the member authorities and the slice authorities. In SFA, the concepts of slices, slivers and RSpecs are defined in the AM API. Slices bundle resources over multiple testbeds, belonging to one (series of related) experiment(s). A sliver is an allocated resource and part of a slice contained to a single testbed. RSpecs are used to specify resources on a single testbed by using or extending previously agreed XML schema documents. Fed4FIRE uses the GENI AMv3 API and is working together with GENI on a Common AM API as a next version. Fed4FIRE therefore inherently adopts the same mechanisms for authentication and authorization based on X.509v3 certificates.

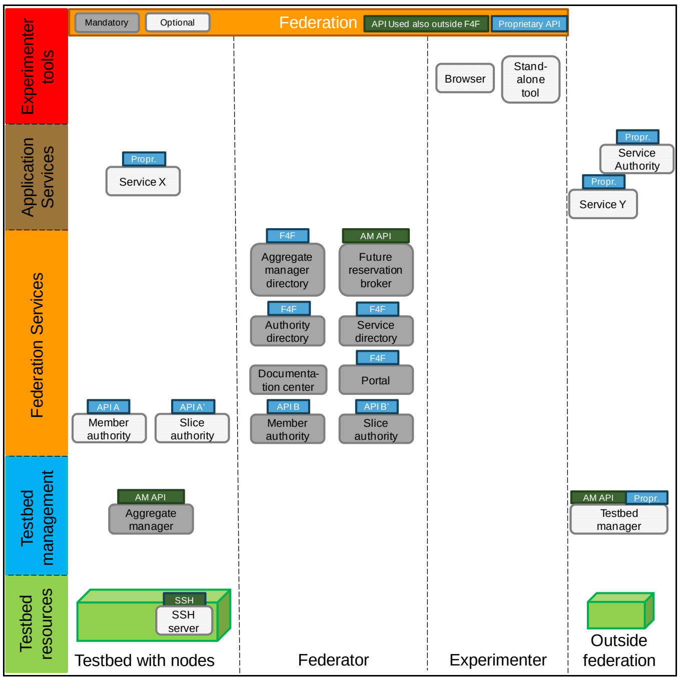

Fig. 2. Architectural components for resource discovery, specification, reservation and provisioning

The federator components for resource discovery, specification, reservation and provisioning include the portal (provides a central overview and registration), the central member and slice authority (allows experimenters to register at the portal, but also other authorities exist at individual testbeds), the aggregate manager (AM) directory (presents an overview of all testbeds' AM contact information), the documentation center (http://doc.fed4fire.eu), the authority directory (signs certificates for authentication/authorization between experimenters and testbeds to create a web of trust), the service directory (lists an overview of federation and

[2] http://www.geni.net

application services) and the reservation broker (resolves abstract reservation requests).

At the testbed side, there are (virtual or physical) resources (often reachable through SSH), the AM (manages the discovery, reservation and provisioning of the resources), the authority (optional) and the application services (if any).

The experimenters can make use of a browser to access the tools hosted centrally or of stand-alone tools[3] that are already installed on the experimenter's computer (e.g. Omni, SFI, NEPI, jFed) and have compatible AM APIs.

D. *Monitoring and measurement*

For monitoring and measurement, the federator components are the FLS dashboard (provides a real-time and comprehensive overview of the health status of the different testbeds, combining facility monitoring with specific measurements), the ORBIT Measurement Library (OML[4]) server and corresponding database (stores the data for FLS and the federation services), the component for nightly login testing (provides an overview of the operational status of the different testbeds, focusing on thorough experiment lifecycle management tests) and the data broker (makes access to experiment data possible through the portal).

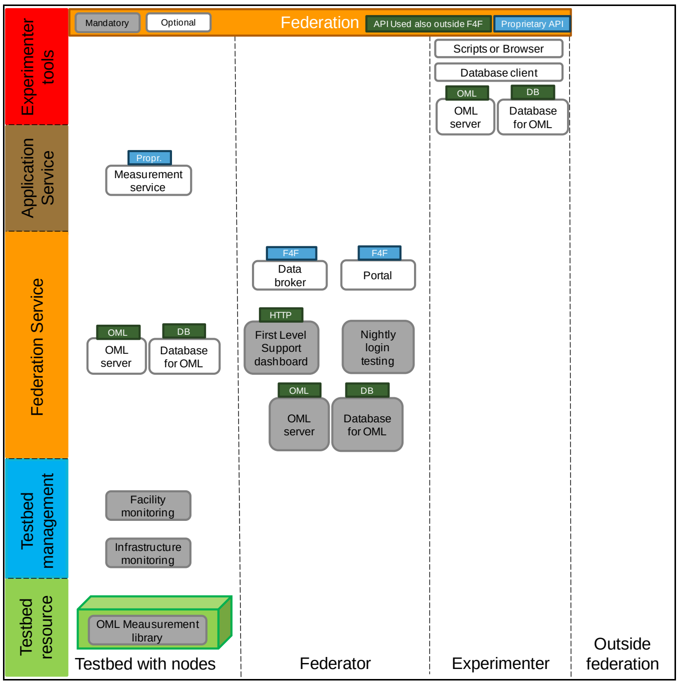

Fig. 3. Architectural components for monitoring and measurement

The components that reside at the testbeds are the facility monitoring component (sends data on health status to the central OML server as an OML stream), the infrastructure monitoring component (provides data on behavior and performance of services, technologies and protocols, that experimenters cannot access themselves directly), the OML measurement library (allows for experiment measuring initiated by an experimenter framework), the OML server (optional, for experimenters to have their own data collection for exclusive use, acts as the endpoint of an OML stream and provides several types of databases) and the measurement service (optional, provides easy access for the experimenter to retrieve data through a proprietary interface).

The experimenters can use any experimentation tools (scripts or browsers) to send queries to the data broker API, a database client to retrieve monitoring and measuring data from any OML server where it was stored or their own OML server and attached database in order to archive the data themselves.

E. *Experiment control*

Experiment control is performed between the experimenter and the testbeds directly, without federator components.

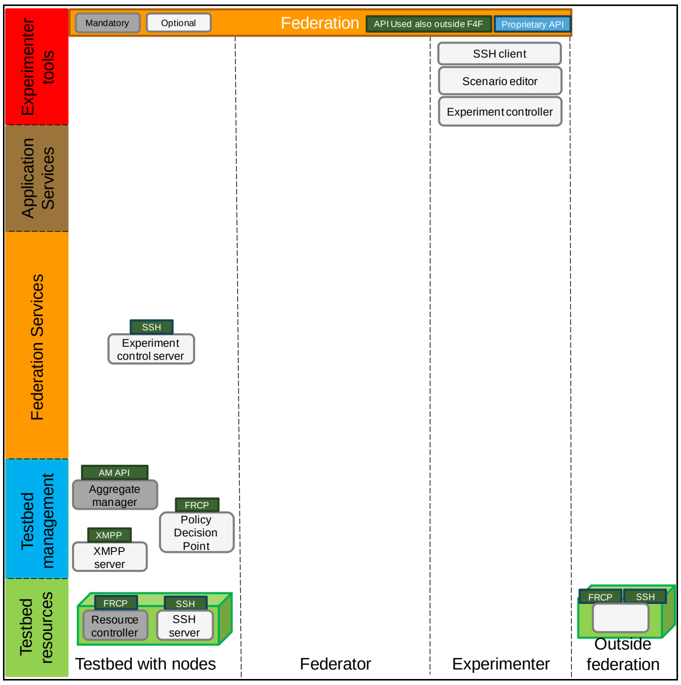

Fig. 4. Architectural components for experiment control

At the testbeds, components for experiment control include the SSH server (for remote access), the resource controller (invokes actions from the experimenter on a (remote) resource using the Federated Resource Control Protocol (OMF FRCP[5]), through an agent), the Extensible Messaging and Presence Protocol (XMPP) server (not mandatory, communicates the FRCP messages), Policy Decision Point (PDP, authenticates and authorizes FRCP messages for the resource controller based on information from the AM) and the experiment control server (processes experiment control scenarios and executes actions by issuing FRCP messages at the correct moments).

At the experimenter side, different tools can be used such as SSH clients (for remote access to testbeds resources and manual experiment control), an experiment controller (similar to the one at the testbed, but deployed locally) and a scenario editor (allows to construct experiment control scenarios).

[3] http://www.fed4fire.eu/tools.html

[4] http://oml.mytestbed.net/projects/oml/wiki

[5] http://omf.mytestbed.net

F. SLA management and reputation services

SLAs are used to manage resources, so users know what resources they are getting from the provider, and the provider can share the resources they have between its users. A typical SLA lifecycle covers the following phases:

- Template specification: describe a clear step-by-step procedure on how to write an SLA template to provide a correct service description.
- Publication and discovery: publish the provider offer, the customer Quality of Service (QoS) needs and the possibility for the customer to browse and compare offers.
- Negotiation: agree on SLA conditions between the experimenter and the infrastructure providers.
- Resource selection: select the resources that need to be assigned to the experimenter to meet the chosen SLA.
- Monitoring and evaluation: compare all the terms of the signed SLA with the metrics provided by the monitoring system, in order to internally prevent upcoming violations and to externally discover potential violations.
- Accounting: invoke a charging/billing system according to the result, make reports and/or invoke a reputation system or an internal policy engine tool.

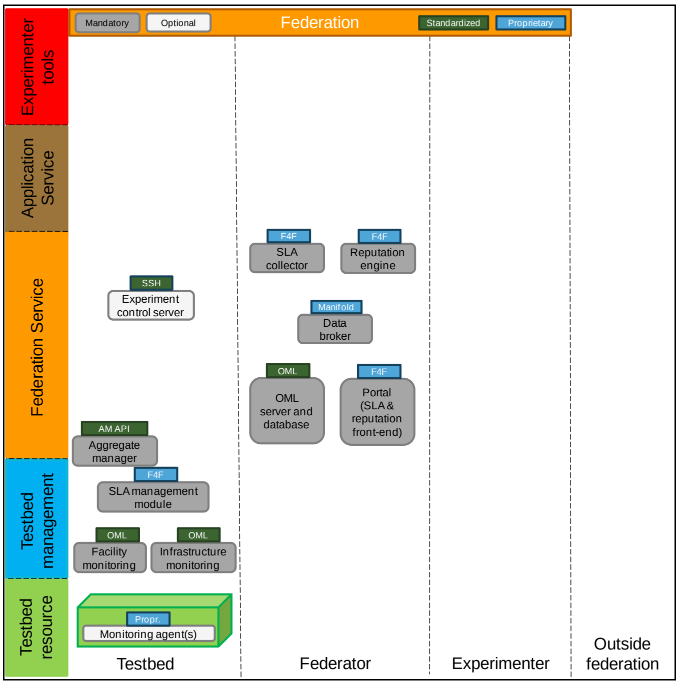

Fig. 5. Architectural components for SLA management and reputation services

There will be one SLA between the experimenter and each different testbed. However, the SLA enforcement will be performed per sliver as an RSpec document, which is delivered by the AM API, determines exactly which resources are available during what time to the experimenter. At the end of the experiment, the global information of the SLAs evaluation of all the slivers of the same testbed for the same experiment will be available. The federator components consist of the SLA collector (acts as a broker that gathers SLA warnings and evaluations and provides SLA information to the reputation service, see below), the SLA management module (supervises agreements by processing and validating the measurements and triggering actions) and the SLA front-end tool (provides a GUI for experimenters, integrated in the portal).

The reputation service aims to provide mechanisms and tools towards building trustworthy services based on the combination of Quality of Experience (QoE) and monitoring data. The developed mechanisms and tools will reflect the experimenters' perspective with the objective of empowering them to select testbeds based on dynamic performance metrics (e.g. testbed up-time, usage, SLA enforcement). These metrics will offer a "smart" user support service that provides a unified and quantitative view of the trustworthiness of a facility. The main interactions of this service with the other components include communication with the reservation broker (to retrieve data regarding the resources used in an experiment), the monitoring data broker (to obtain measurement data for the reserved resources), the SLA service (to obtain SLA information regarding violations) and the portal (serves both as a place for displaying testbeds' reputation scores and as a feedback page for users' QoE).

IV. Conclusion

This paper provides an overview of the Fed4FIRE federation architecture, designed based on a comprehensive set of requirements from different sources and communities. For each of the steps in the experiment lifecycle, architectural components with their interfaces and interactions have been presented. This architecture evolved from its earlier version from cycle 1 [1], with major improvements and clarifications. We expect the final version (in cycle 3) to further develop towards more unified interfaces, taking feedback from the different Open Call experimenters into account.

Acknowledgment

This work was carried out with the support of the Fed4FIRE project ("Federation for FIRE"), an integrated project funded by the European Commission through the 7th ICT Framework Programme (318389).

References

[1] W. Vandenberghe, et al., "Architecture for the heterogeneous federation of Future Internet experimentation facilities", Future Network and Mobile Summit, Lisboa, July 2013, pp. 1-11.

[2] B. Vermeulen, et al., "Second federation architecture", available for download at http://www.fed4fire.eu/fileadmin/documents/public_deliverables/D2-4_Second_federation_architecture_Fed4FIRE_318389.pdf, last accessed May 16, 2014.

[3] L. Peterson, R. Ricci, A. Falk and J. Chase. "Slice-Based Federation Architecture", available for download at http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf, last accessed May 16, 2014.