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

D2.13: Testbed requirements, developments, and integrations – Final version

Work package	WP 2
Task	Task 2.2
Due date	31/12/2021
Submission date	8/03/2022
Deliverable lead	Imec
Version	4
Authors	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Albert (Yiu Quan) Su (SU)
Reviewers	Peter Van Daele (imec), Maria Chiara Campodonico (Martel)
Abstract	This deliverable describes the specific requirements, developments and integrations that were done in the first 48 months to add new testbeds to the federation.
Keywords	Testbed integration, federation, testbed support

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
V1	30/10/2021	TOC	Brecht Vermeulen (imec)
V2	8/03/2022	First complete version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Albert (Yiu Quan) Su (SU)
V3	15/03/2022	Submitted version	Peter Van Daele (imec)

DISCLAIMER

COPYRIGHT NOTICE

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	**X**
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.

EXECUTIVE SUMMARY

This deliverable is the 3rd deliverable in a series describing the specific requirements, developments and integrations carried out to integrate new testbeds into the Fed4FIRE+ federation. This 3rd deliverable provides a complete overview since the start of the project.

In this period, in total 30 testbeds were added to the federation requiring 15 different types/implementations. Since the production of the previous deliverable in this series, deliverable D2.9, 2 testbeds have been added, but by the end of 2021 all Exogeni and FUTEBOL testbeds were removed as they were end-of-life.

About 60 testbeds were removed from Fed4FIRE over time because the testbed was end-of-life. Today about 65 testbeds remain federated. This shows that it is very important to support the concept of 'testbeds, tools and authorities come and go over time'.

The two sections with feedback from the testbed providers on the experience and learnings they gather from open call experiments running on Fed4FIRE were updated.

The federation template that we support together with Geni (Geni Control Framework, GCF) has been further extended, a.o. with general GDPR support. This results in a Docker AM which is fully functional for docker resources (including IPv6).

This deliverable is a full update of D2.9 (and thus includes all info of D2.4 and D2.9 as a complete reference). A section has been added with information how to join the Fed4FIRE federation in a nutshell.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

This deliverable is the 3rd deliverable in a series describing the specific requirements, developments and integrations carried out to integrate new testbeds into the Fed4FIRE+ federation. This 3rd deliverable provides a complete overview since the start of the project.

In this period hs, in total 30 testbeds were added to the federation requiring 15 different types/implementations. Since the production of the previous deliverable in this series, deliverable D2.9, 2 testbeds have been added, but by the end of 2021 all Exogeni and FUTEBOL testbeds were removed as they were end-of-life.

About 60 testbeds were removed from Fed4FIRE over time because the testbed was end-of-life. Today about 65 testbeds remain federated. This shows that it is very important to support the concept of 'testbeds, tools and authorities come and go over time'.

This deliverable is made up of 5 parts:

Section 2 provides an overview of the testbeds which have joined the federation since the start of the project. Most of them are external testbeds, but some of the testbeds are new testbeds at Fed4FIRE+ partners or further integrations of the testbeds at the Fed4FIRE+ partners.

All of the testbeds at the Fed4FIRE+ partners provided feedback on how Fed4FIRE+ and belonging to the federation meant added value to them. This information is described in section 3.

But the testbeds also "listen" to their users and suggestions / comments made by individual experiments from the Open Calls which triggered specific actions / modifications / updates at the testbeds are also provided in detail in section 4.

The federation template that we support together with Geni (Geni Control Framework, GCF) has been further extended, a.o. with general GDPR support. This results in a Docker AM which is fully functional for docker resources (including IPv6) and described in section 5.

This deliverable is a full update of D2.9 (and thus includes all info of D2.4 and D2.9 as a complete reference). A section 6 has been added with information how to join the Fed4FIRE federation in a nutshell.

2 TESTBED FEDERATION EFFORTS

2.1 GENERAL DEVELOPMENT FOR FEDERATING TESTBEDS

The documentation aimed at assisting testbed owners when federating their testbeds, was extended on many points, based on feedback and questions from the testbeds. All this documentation was bundled in the updated Fed4FIRE documentation website and can be found at: https://doc.fed4fire.eu/#testbed-owner-documentation.

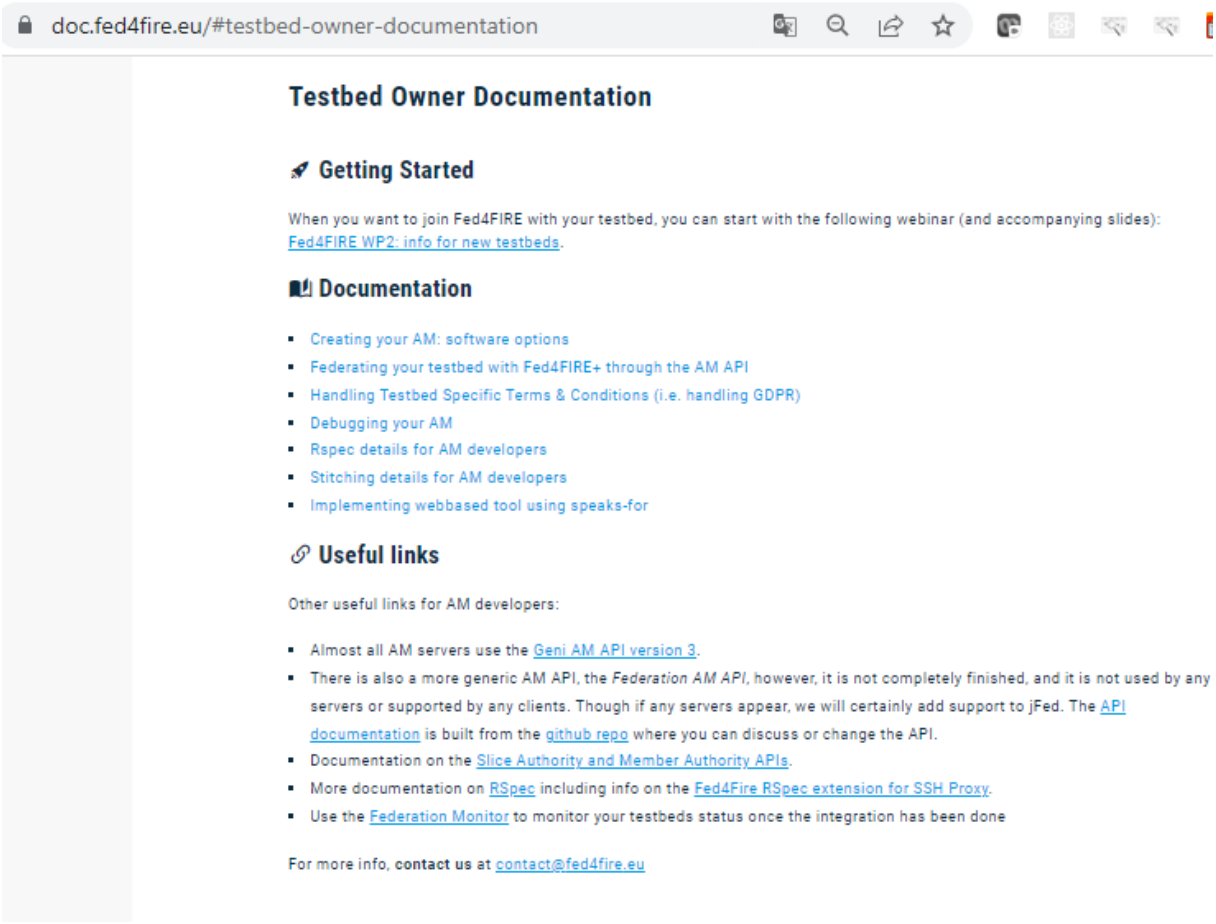


Figure 1: Overview of Fed4FIRE documentation for testbed developers

Inside the jFed experiment GUI tool, support for different link types was extended to manage the needs of the Futebol testbeds. jFed will now automatically select the right type of link, based on which nodes (of which testbeds) are connected with each other. This makes it much easier for users, and reduces the need to know all technical details about some testbed links.

Figure 2 gives an overview of the current support resource types (icons on the left), and examples of the new types in the canvas at the right.

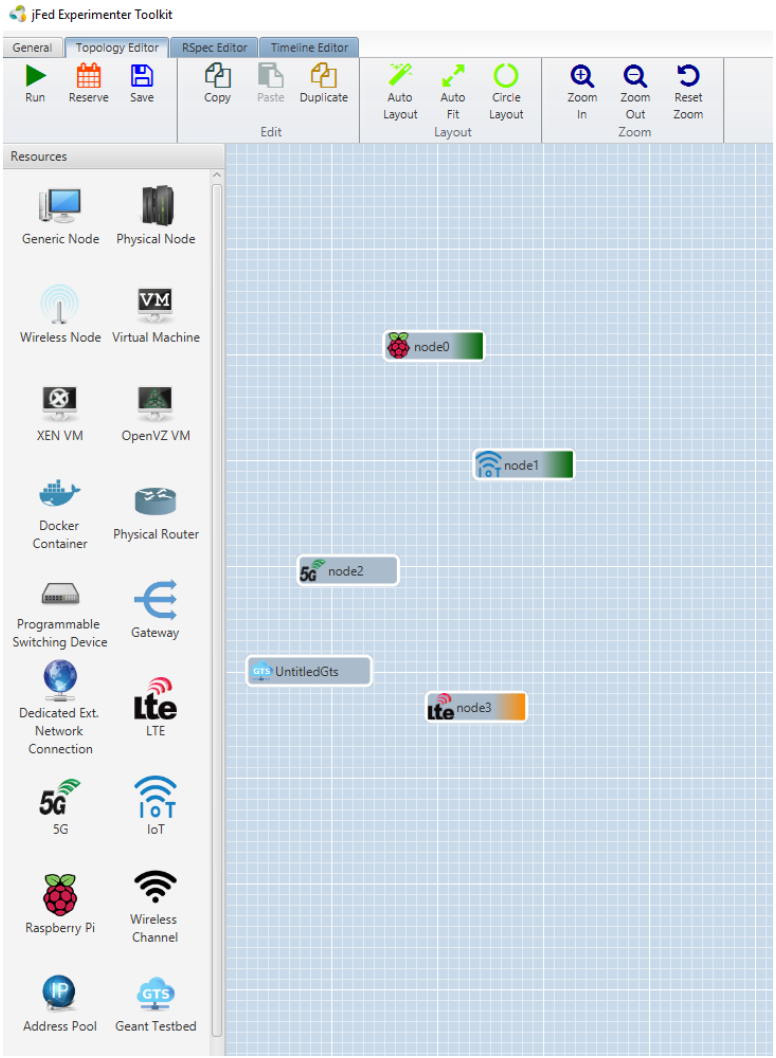


Figure 2: Overview of jFed resource icons, including new icons for Raspberry, 5G, IoT and GTS (Geant testbed as a service)

2.2 FEDERATED TESTBEDS

2.2.1 FUTEBOL Brazil/UFES

The FUTEBOL testbed of the Federal University of Espírito Santo was federated during April 2017 to Dec 2017. This testbed offers access to VMs. A lot of assistance was required, but no new jFed developments were needed.

2.2.2 FUTEBOL Brazil/UFMG

The FUTEBOL testbed of the Universidade Federal de Minas Gerais was federated begin 2017. This testbed offers access to a wide range of experimental hardware: USRP (software defined radio) hardware, TelosB sensors, bare metal wifi nodes, and bare metal Raspberry Pi nodes.

jFed was extended with more complex support for hardware and sliver types. This allowed adding more flexible "icons" in the jFed GUI. A Raspberry Pi and an IoT icon (Figure 3) were then added and linked to the appropriate configuration of the UFMG testbed.

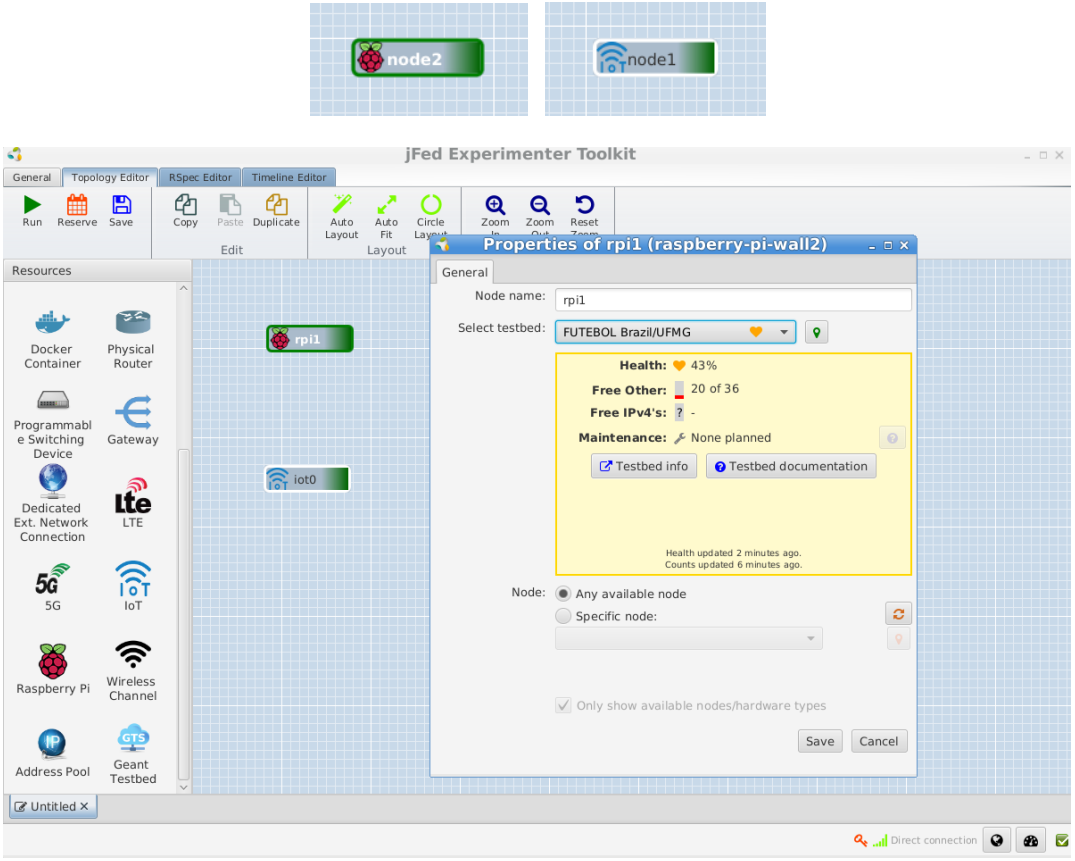


Figure 3: Raspberry Pi (top left) and an IoT icon (top right) added to the jFed GUI

2.2.3 Futebol Brazil/UFRGS

The FUTEBOL testbed of the Federal University of Rio Grande do Sul was federated begin 2017. This testbed offers access to USRP (software defined radio) hardware, VMs, and bare metal Raspberry Pi nodes.

The federation work required for FUTEBOL UFMG was developed in parallel for this testbed, which had the same requirements. The new Raspberry Pi icon in jFed was linked to the appropriate configuration of the UFGRS testbed.

2.2.4 FUTEBOL VTT

The FUTEBOL VTT testbed of the Technical Research Centre of Finland was federated in April 2018. Little federation support was needed, and no new jFed development was required.

2.2.5 OWL TUD

The Online Wireless Lab (OWL) testbed of the Technische Universität Dresden was federated in March 2018. A lot of testbed side debugging was required.

The biggest issue turned out to be the very long image load time. jFed was modified to handle long load times in a more user-friendly way.

A 5G icons was added to jFed, and the testbed was linked to this icon.

Figure 4: The new icons in jFed, including the 5G icon

2.2.6 ARNO Sant'Anna Pisa Testbed

The ARNO Testbed at Sant'Anna Pisa was federated between June and Aug 2017. The testbed is based on the docker AM, and it took some testbed side debugging to get it federated. The testbed offers access to LTE equipment. No jFed development was required for this tested.

2.2.7 FIT R2Lab

The R2Lab testbed of FIT was federated May 2018. This testbed allows access to wireless hardware. No new jFed development was needed, and the testbed was added under the 5G icon.

2.2.8 ESAT MM Testbed

The ESAT Massive MIMO Testbed of KU Leuven was federated between October and December 2017.

We helped develop the modified GCF AM for this testbed, as some complex features were needed. jFed was also extended to support some of these features, in particular, support for connecting to windows nodes using RDP was added, and support for working with gateway nodes added by the testbed (not requested by the user) was added. The testbed was added under the 5G icon in jFed.

2.2.9 Imec City of Things Antwerp

The City of Things testbed in Antwerp was federated. This testbed required little federation work. Because the testbed uses special links by default (gre links), jFed was extended to clearly differentiate between different link types in the GUI.

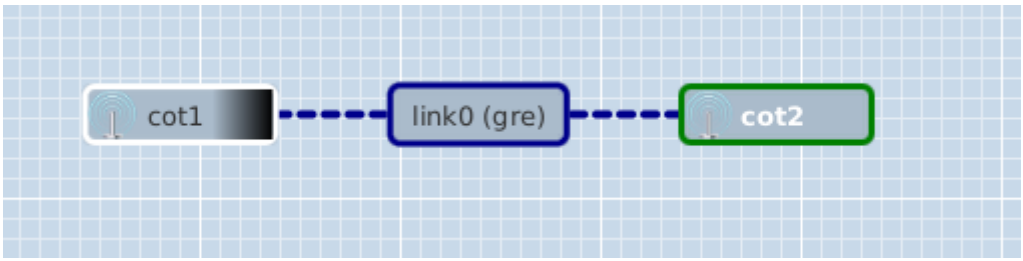


Figure 5: A link between City of Things nodes in jFed

2.2.10 Fuseco removed (new since D2.4)

The Fuseco testbed of Fraunhofer has been removed from the Fed4FIRE federation as Fraunhofer does not allow remote access anymore.

2.2.11 Imec GPULab (new since D2.4)

The imec GPULab testbed (https://doc.ilabt.imec.be) has been opened for public in 2019. GPUs were already available in Virtual Wall nodes (and accessible through jFed), but the problem was that all needed software (CUDA drivers and libraries, machine learning frameworks, etc) had to be installed by the users and this was a burden for the machine learning experts. GPULab answers those needs and can be accessed in three different ways:

- Interactively through a Jupyter notebook (https://jupyterhub.ilabt.imec.be)
- Website access to launch GPULab jobs (mostly used for newcomers) (https://gpulab.ilabt.imec.be/)
- Command Line launching of tools through the GPULab-cli tool (https://doc.ilabt.imec.be/ilabt/gpulab/overview.html#command-line-interface)

For the web-based access, the newly added OAuth functionality of the new user portal is used (see D3.4), so users can login web-based with a single sign on.

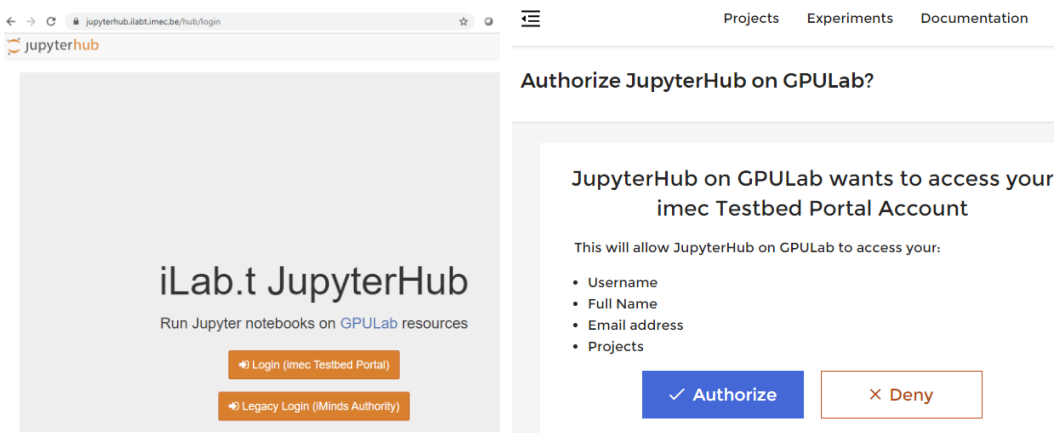


Figure 6: Single-sign-on for GPULab through OAuth

For the command-line, we use the same X.509 certificates (PEM file) as in jFed, see https://doc.ilabt.imec.be/ilabt/gpulab/cli.html#prerequisites. These are used to submit jobs and to login interactively with ssh.

2.2.12 Inria Grid5000 (new since D2.4, updated since D2.9)

Grid5000 has now also been federated (although not all native functionality is supported through the AM API). The following work has been done to support this:

- Support in mapping AM interface to grid'5000 testbed (by clearing up misconceptions and proposing ways to implement certain calls)
- Help in debugging some AM implementation issues
- jFed missing feature added: "testbed proxy" support for ESpec
- Added development and production server of grid'5000 to jFed config
- Added jFed support for grid'5000 AM unique behaviour (= extra feature and workaround flags added and supported):
- No slice and ssh sharing is allowed by grid'5000: jFed now knows this, and takes it into account.
- grid'5000 AM only supports the user SSL SSH keys, not the full per slice flexibility normally provided by Provision call. jFed now knows this, and takes it into account.
- Multi site is reflected in the URNs: basic jFed support is implemented, more advanced support (= selecting site inside jFed GUI) is to be added.

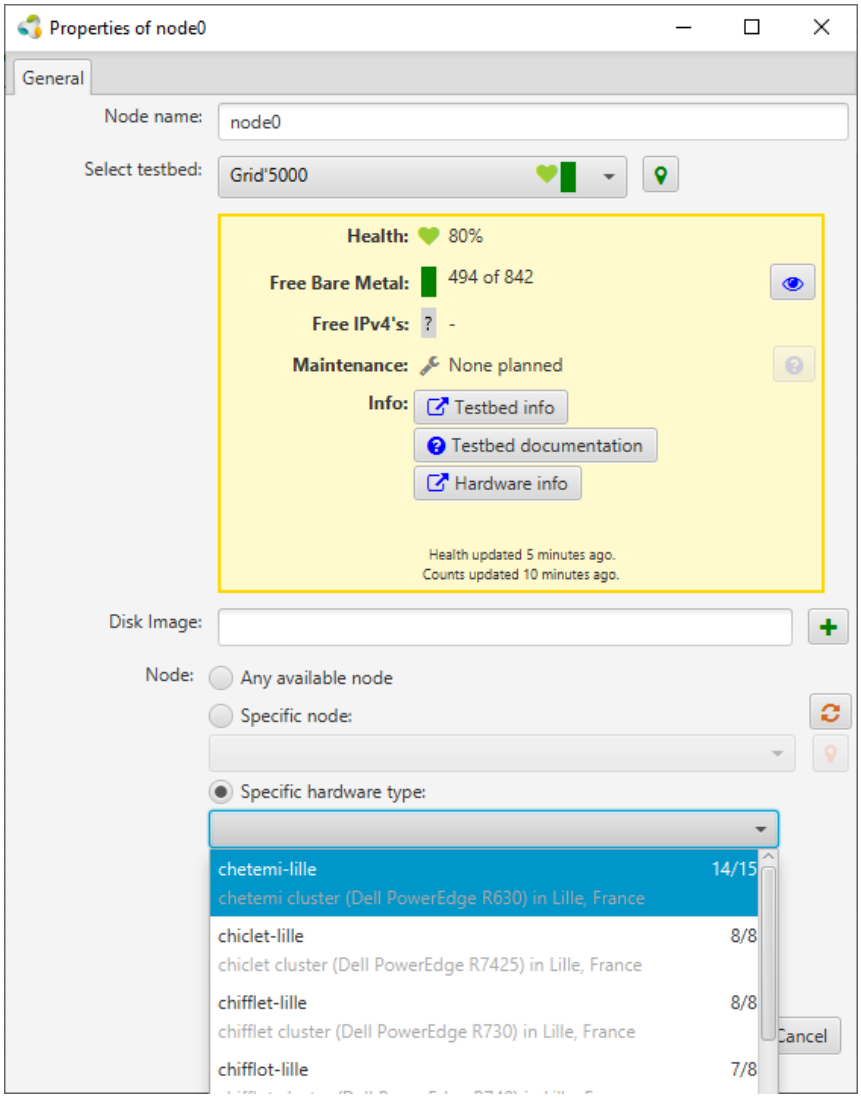


Figure 7: Part of the hardware types for GRID5000

Since D2.9 we also added the following feature: The Inria Grid 5000 testbed is a complex testbed with multiple locations in France, so we had to add extra logic to jFed so the users can choose the right location or hardware type of nodes, depending on their type of experiment. This can be seen in the screenshot below (Figure 8) where a user wants to select any hardware but located in Lyon.

The Inria Grid 5000 people have documented their lessons learned at https://hal.inria.fr/hal-02962845.

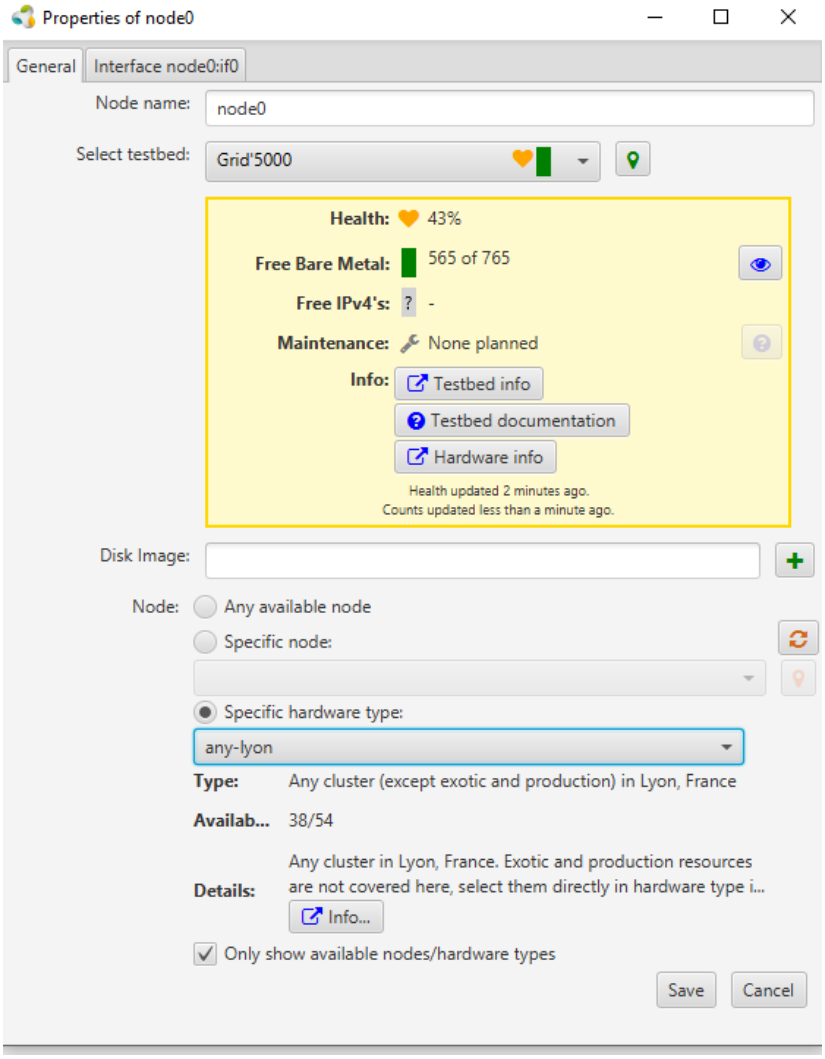


Figure 8: Screenshot of the jFed GUI allowing to choose the location of the hardware in the GRID5000 testbed.

2.2.13 Mandat International IoTLab (new since D2.4)

The support for this AM was very straight forward and nothing had to be changed to jFed, apart from adding the testbed info and helping to debug.

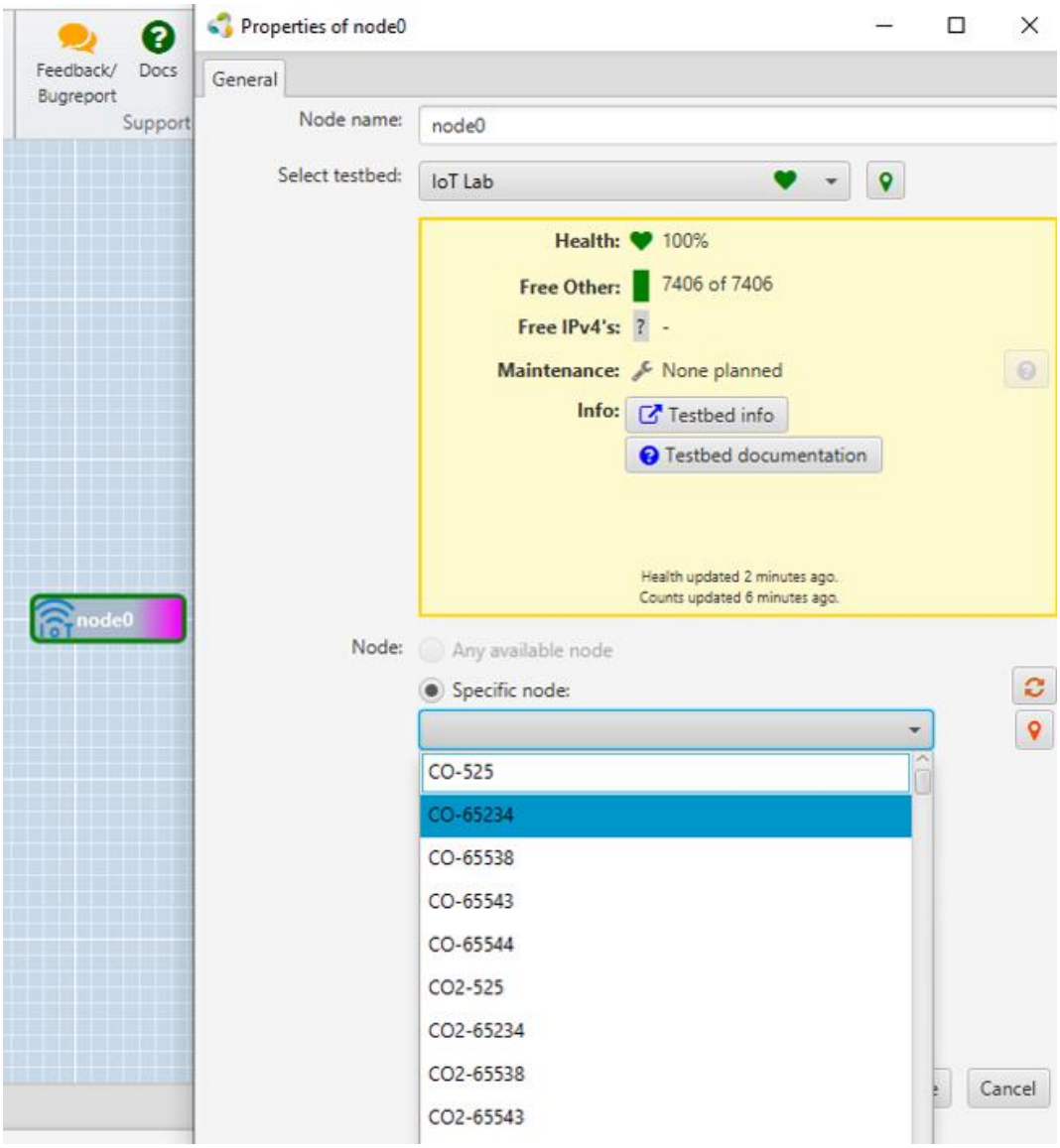


Figure 9: Choosing a node for IoTLab

2.2.14 IRIS testbed (updated since D2.9)

For the Iris testbed (which is now openstack based) it is possible to choose the flavour of the virtual machine to be used, so we had to add that as well in the node properties as can be seen below in the jFed screenshot.

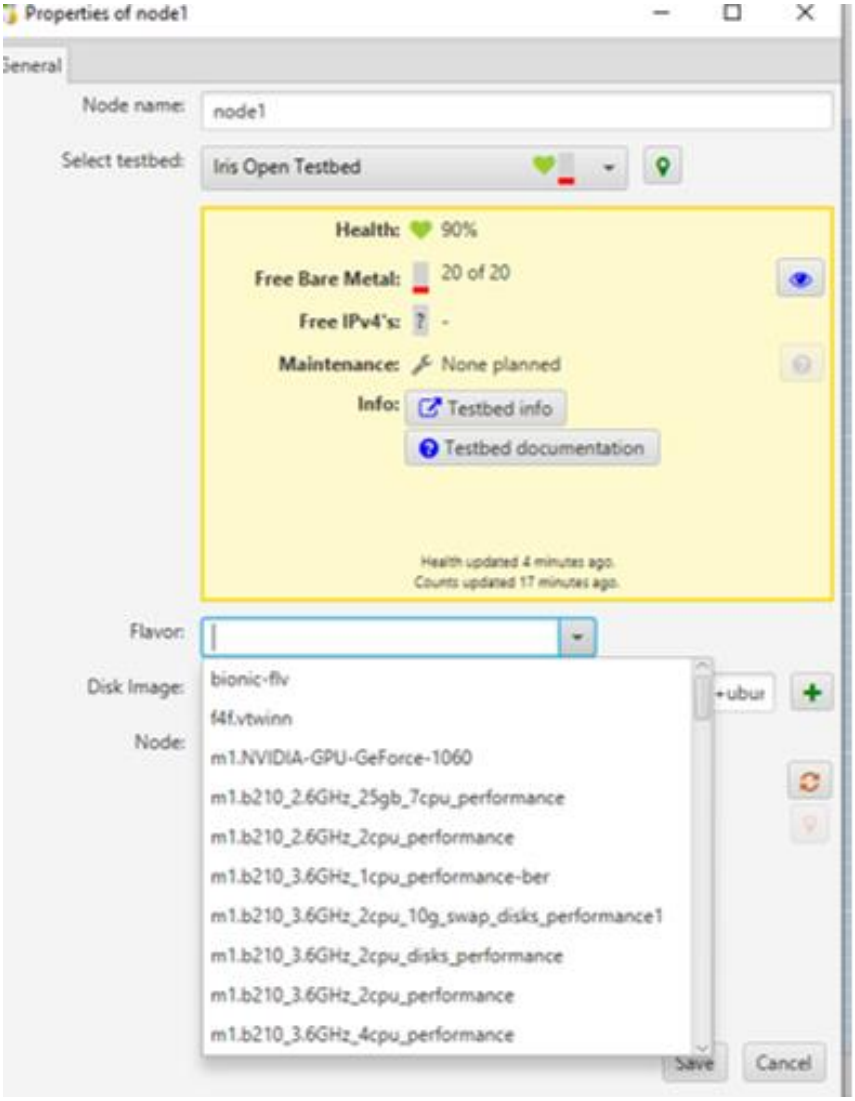


Figure 10: Option in jFed GUI to choose the flavour of the virtual machine in the IRIS Testbed

2.2.15 Other testbeds

Two development and test servers were added (for University Bristol and UFMG). These are not production testbeds that can be used by federation users.

A lot of new US ExoGeni and InstaGeni testbeds were federated. Adding these to the federation required almost no effort:

- ExoGENI CIENA HQ
- InstaGENI ODU
- InstaGENI Hawaii
- InstaGENI VT
- InstaGENI UVM
- InstaGENI Louisiana
- InstaGENI UTDallas
- InstaGENI UCSD
- InstaGENI Utc
- InstaGENI University of Washington
- InstaGENI Colorado
- Exogeni UNF (new since D2.4)
- Exogeni LAT (new since D2.4)
- Instageni VCU (new since D2.4)
- Instageni ODU (new since D2.4)
- 2 exogeni testbeds (new since D2.9)
- All exogeni testbeds and FUTEBOL testbeds were removed at the end of 2021 because they were end-of-life (new since D2.11)

2.2.16 Associated testbeds

One testbed has been added since D2.4 as associated testbed: 5G INNOVATION HUB NORTH run by Luleå University of Technology together with Ericsson, Telia and Tieto.

See https://www.fed4fire.eu/associated-testbeds/

The difference between advanced and light federation and associated testbeds is documented at https://www.fed4fire.eu/add-your-facility/.

- Associated: the testbed is listed on the website, with links to contact information and documentation. This option does not require technical integration.
- Light: access to the testbed's resources is realized by exposing a Web-based API. This option does not allow full control over the individual testbed resources, but ensures unified access to experimenters.
- Advanced: the testbed is fully integrated in the federation so that experimenters can interact with their experiment during all stages of the experiment's life cycle (resource selection, instantiation, control, monitoring, etc.). This option requires the implementation of the Federation AM API (Aggregate Manager Application Programming Interface) on top of your testbed.

Some testbeds might be included under advanced federation for usage of general resources, while specific features, such as proprietary services, are exposed through light federation.

3 ADDED VALUE FOR FED4FIRE+ TESTBED PROVIDERS (updated since D2.9)

For each of the testbeds, a short overview is presented below on what is perceived as added value by the testbed owner on the participation of the testbed in the Fed4FIRE+ program. This is updated compared to D2.9.

3.1 TESTBED CITYLAB (IMEC)

Thanks to Fed4Fire+, the testbed has been improved to meet the demand of research experiments. In addition, this allows us to discover new use cases that can be applied to the use case. The feedbacks received from the open call experimenters have also contributed to having the hardware usage better managed. In the case of smart highway experiment, experiences from the experiments have also triggered us to developed more standardized technologies deployed in the testbed.

The most important feedback we got from Fed4FIRE+ can be summarized into three themes:

- Technologies: CityLab started by focusing solely on unlicensed technology in the city. While executing our experiment, we noticed the need for LTE and C-V2X. The step to adding these technologies was relatively simple (through a combination of an experimental license and a collaboration with the portable testbed for LTE, and opening up our smart highway testbed for C-V2X). This kind of feedback would have been very hard to get upfront from a general audience.
- Locations: experimenters gave us critical feedback on the number of gateways (good), the locations (denser needed) and the relative positioning (more line of sight needed). This kind of feedback was hard to assess upfront; we are very happy with this kind of external validation and feedback.
- Usage: our testbed is oriented very broadly to smart city applications. With the open calls, we were able to provide more specific use cases and share experimenter happiness (which was high). This became a strong instrument in attracting further (industrial) interest in our infrastructure.

3.2 TESTBEDS VIRTUAL WALL, W.ILAB-T, PORTABLE TESTBED, GPULAB, TENGU (IMEC)

Through the Fed4FIRE+ federation, we have been able to offer a much better toolset to the users (jFed, new portal, Federation monitoring, international testbed access, automation tools, GPULab jupyter notebooks). We have also a louder international voice as the federation of heterogeneous resources has a bigger international impact. We get a larger variety of experiments on our testbeds (and new users) and as such need to extend the feature set and resources, from which our internal users can benefit then as well. By having completely new users (in comparison e.g. to new imec students which get introduced by imec professors or seniors), we need to also keep our documentation up to date which is to the benefit of everyone. Without Fed4FIRE+ the tools (jFed, federation monitor, portal) would be less rich in features and less mature. The international connectivity (within Europe, but also to US, Brazil, Japan) would not have been established without Fed4FIRE. Finally, being part of a larger federation makes also that our internal users can use other testbeds, e.g. if we do not have the right resources, or if we do not have the resources available to them.

3.3 TESTBED EDGENET (SU)

Fed4Fire+ contribution to the EdgeNet testbed is twofold. First, it gives credibility and visibility to the project, allowing it to attract new experimenters. Second, the experiments ran through Fed4FIRE+ allows to discover new use cases for the testbed, and to make the appropriate improvements.

3.4 TESTBED NITOS (CERTH)

Through the participation in Fed4FIRE+, NITOS gained visibility and more experimenters through open access and open calls. It should be mentioned that CERTH during M24-M48 supported at least 7 opencall experimenters and 19 in total since the beginning of the project. Being an active member of the federation, it enables NITOS to follow the latest technology trends in testbeds and federation tools and services. Moreover, feedback received by the opencall experimenters allowed us to improve specific aspects of our testbed, which we had overlooked. These include, not updated documentation on recent changes of the testbed and its tools, misbehavior of certain resources and testbed's software tools bugs.

3.5 TESTBED IOT LAB (MANDAT INTERNATIONAL)

As the IoT Lab testbed was integrated to the Fed4FIRE+ testbed federation in the previous reporting period, IoT Lab was listed as a testbed available for the Open Calls. Several companies contacted Mandat International to get more information about the possibilities of tests provided by the IoT Lab testbed. Furthermore, this integration permits to the IoT Lab users, typically the researchers, to access the resources of Fed4FIRE+ which are of course completely different from the resources offered by IoT Lab. As the same technologies is used by Fed4FIRE+ and IoT Lab, there are several synergies and collaboration between the IoT Lab and Fed4FIRE+ testbeds.

3.6 TESTBED NETMODE (NTUA)

As Fed4FIRE+ testbed providers, we patroned one open call experiment during the reporting period. The feedback of these experiments was valuable to modify and extend our testbeds. Furthermore, being testbed provider of Fed4FIRE+ consortium improve the visibility of our testbed and allows us to meet researchers from both academia and industry and establish new collaborations.

3.7 TESTBED GRID5000 (INRIA)

At Inria we see our involvement within Fed4FIRE(+) as a way to build scientific and technical collaborations within the European testbeds community. While we are a fairly established testbed in France, we lacked the high-communication-bandwidth interactions with other strong actors in the European scene that this project provides.

Fed4FIRE+ also forces us (in a good way) to reconsider our design choices and compare them with those made by other infrastructures, leading to building better technology overall. This is outlined in the work done in T3.2 during this reporting period: Bridging the GENI/Fed4FIRE+/SFA world, and the design choices of Grid5000, led to many interesting challenges that resulted in a scientific publication describing our work.

3.8 TESTBED I2CAT OFELIA (I2CAT)

The participation of the testbed in Fed4FIRE+ federation provides an extra value regarding the (i) visibility of the testbed to the outer world; the (ii) adoption of procedures regarding the data processing operations and the compliance with GDPR; as well as the (iii) integration with tools that monitor the status of the testbeds and resources offered by it.

3.9 TESTBED SMARTSANTANDER (UC)

Fed4FIRE+ increases the visibility of our testbed on domains which are not usually targeted by us. This contributes to a higher interest and usage of our platform, hence to a better sustainability of SmartSantander on the international research ecosystem.

SmartSantander is federated following the service-oriented architecture approach (as opposed to the SFA/jFed one), so possible technical improvements on the federation tools during Fed4FIRE+ could enable new inter-domain experimentation which can be adressed by using a single testbed.

3.10 TESTBED TRIANGLE (PERFORMLTE) (UMA)

Due to our participation in Fed4FIRE+ UMA has been able to contact with multiple organizations that have become partners in multiple research projects, which in turn enabled us to improve the equipment and features provided by our testbed. In addition to this, the participation on Open Calls allowed us to validate our infrastructure and workflow, while identifying possible improvements.

3.11 TESTBED EXOGENI (UVA)

Through our participation in Fed4FIRE+ our testbed gets added exposure, more people become aware of its features. The second advantage is that by interacting with the community we can constantly maintain our setup to be state-of-the-art, for example.

3.12 TESTBED LOG-A-TEC (JSI)

By participating in Fed4FIRE+, we have increased the international visibility of the Log-a-Tec testbed, reaching a wider audience and thus attracting additional attention to our work, which also leads to an increase in new experiments through open calls and open access. The various recommendations and requests from experimenters drive the development and upgrades of the testbed. The feedback allows us to modify the testbed, expand it, add missing functionalities, validate new features, and identify usability improvements that ensure the sustainability of the testbed.

The testbed continuously enables to meet researchers, establish new collaborations, and strengthen existing ones. In addition to the open call experiments, we managed to win the project with the University of Montenegro and a national research project, extend the collaboration with the University of Banja Luka and use the functionality of the testbed in H2020 projects.

3.13 TESTBED IRIS (TCD)

The Fed4FIRE+ project has been instrumental in enabling CONNECT at Trinity College Dublin to participate in large international cooperative research projects and support direct collaborations with other research groups nationally and internationally of high repute. For example, the infrastructure offered by the Iris testbed, which is supported by Fed4FIRE+ resources and tools, has acted as the foundation for many successful FP7, H2020 projects, Next Generation Internet (NGI), and Science Foundation Ireland funded projects that CONNECT has been involved in. These include projects related to advanced experimentation and research in areas such as radio (CREW, ORCA, WishFul, eWINE, FUTEBOL), artificial intelligence (SATORI), software defined networking and optical communication (FUTEBOL), cloud (Fed4FIRE+, Fed4FIRE, and 5GINFIRE), and remote experimentation and education (FORGE).

The contacts and relationships developed supporting Fed4FIRE+ open call experimenters, attending FEC conferences, and interaction with Fed4FIRE+ consortia has encouraged and expedited new project collaborations and consortia with CONNECT for proposals across H2020, 5GPPP, and EU-Brazil initiatives, and collaborative EU-US Next Generation Internet Experiments. For example, most recently the CONNECT centre has participated in the NGI Atlantic call as an open call winner, and are now cooperating very successfully with Columbia University

and Rutgers University in the USA in the optical-wireless research area. A fundamental component of this experiment and its success is open access and experiment reproducibility which is supported by the Fed4FIRE+ jFed framework. We are also working on collaborations with Virginia Tech in the cyber security space, and Rutgers University with a humanities project involving geographically distributed artists working collaboratively in parallel. These recent successes and collaborations can be directly attributed to the Fed4FIRE+ project.

Furthermore, Fed4FIRE+ funded open calls have acted as an excellent starting point to support collaborations with Irish and EU SMEs including BELETEL, Software Radio Systems (SRS), ALLBESMART, etc. For example, ALLBESMART has provided critical support to the Iris testbed team with regards to the extension of the testbed to support Open Air Interface 5G. Furthermore, BELETEL are now providing outdoor OpenRAN radio equipment to support the outdoor 5G testbed at Trinity College Dublin. TCD is also very involved with the Software Radio Systems (SRS) team researching and testing the development of the srsRAN framework. We are investigating collaborations with some of the other SME entities funded to use the Iris testbed via Fed4FIRE+ including the Italian SME SMA-RTY. This is due to their expertise with GPU and FPGA processing and PHY and MAC layer protocols.

The continuously evolving Fed4FIRE+ toolset, the various demands and recommendations from open call experimenters and researchers are driving the evolution and upgrade of our testbed. In 2020-2021 this has involved the extension of the testbed to support live streaming from HD camera's, two GPU units supporting ML training, and extensive work in open source 5G technologies including OpenAirInterface 5G and srsRAN. We see these requirements as fundamental to support our future research goals, objectives, and education activities at Trinity College Dublin. We expect these upgrades to the Iris testbed will form a solid foundation for future proposals at a national level (SFI) and international (EU and US) such as Horizon Europe.

4 IMPROVEMENT ON FACILITIES / TOOLS IMPLEMENTED

The improvements of 2020-2021 have been added to this section and indicated as such (new since D2.9).

4.1 TESTBED CITYLAB (IMEC)

We have added the smart highway C-V2X testbed, which is closely related to the CityLab testbed, as part of our CityLab offer. This directly answers open questions regarding vehicular networks, and allows us to validate new infrastructure.

Moreover, we have requested an LTE testing license, to further enable LTE testing and collaborations with the portable testbed.

Finally, we have also added ath9k WiFi cards, to support low-level IEEE 802.11 experiments.

New since D2.9: The management of multiple users on the testbed has been improved. Learning from having experiments with multiple parties has taught the lesson of better administering the use of the nodes of the testbed. In addition, the GNSS receptors has been replaced on the testbed on the highway. This way, the time correction was greatly improved resulting on the correct and timely message transmission and reception. Moreover, we have also developed apps that can accommodate the use of the testbed in the highway scenario. Experimenters can now use the testbed using standardized message with the new apps installed and integrated in the testbed to test experiments close to real-life scenarios.

4.2 TESTBED VIRTUAL WALL (IMEC)

4.2.1 EXPERIMENT: Stress-test Asvin.io (Asvin)

Quote from report:

"The Fed4Fire+ experiment helped us to validate our product for the market fit. Our expectations have been outperformed. Great experiment facilities and helpful insights. Thanks for the Support to all Members of Fed4Fire+."

Response:

- They used an automatic docker setup (we used internally already) and scaling where we helped with support.
- this will be added to the public documentation website

4.2.2 EXPERIMENT: IIoT-REPLAN (Queens univ of Belfast)

Quote from report:

“We think that the following would be really nice to have and in order to enhance the overall Fed4FIRE+ experimentation experience: A more user-friendly interface for uploading/downloading files to/from the reserved nodes.”

Response:

- IPv6 at client side will solve this as then you can use all kind of standard scp transfer tools. (as our nodes do not have a public IPv4 address by default).
- To help this (without IPv6), we have added a very basic upload and download possibility in jFed which this experiment used.
- In the meantime, it is possible to upload and download files through a jupyter notebook web interface which is very user-friendly (https://doc.ilabt.imec.be/ilabt/gpulab/storage.html#access-from-jupytherhub)

4.2.3 EXPERIMENT: ERASER (UPC)

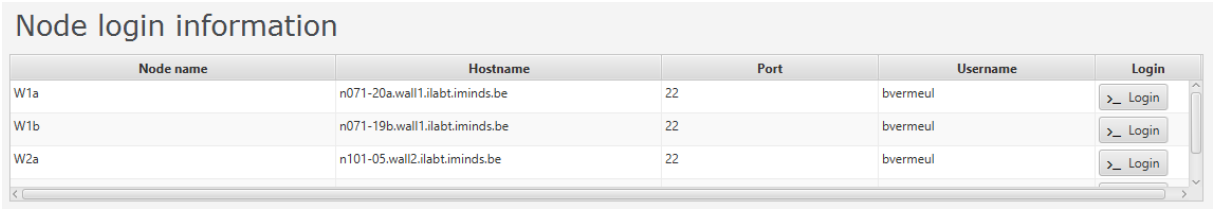
Quote from report:

> A functionality that could be great to add, if possible, is an experiment queue, allowing to schedule an experiment execution, once any (or several) of the current ones finish. In this way, if the duration of the experiment currently running would be known, the system could be able to give a prediction about the time when the scheduled experiment will start and finish.
>
> Regarding JFed, it might be appropriate to include a node searching operation given the name of the physical machine (e.g., n061-02b.wall2.ilabt.iminds.be) where it is allocated at the experimentation facility. Otherwise, it can be difficult to find it in JFed among a large number of nodes. Moreover, it would be interesting to display information about the image that has been loaded in a node in the JFed GUI, once the scenario is setup and running (in a scenario where different images are installed at network nodes).

Response:

- About queuing of the experiments: this is done for GPULab where the experiments are typically more job oriented. For the virtual wall, there is not really a demand for this by other experimenters because experimenters can typically run their experiment without any problem. This experiment used larger topologies and there we advise currently to contact us. In w-Ilab.t there is a reservation system, but this has also drawbacks (people are just not finished with their experiment and the slot is over, or they reserve too much time). For the Virtual Wall the current allocation system is the one comfortable for most (>95%) of the users.
- jFed find a node: this is available in the RSpec view and through the export as node login info (csv so easy searchable), see screenshots below (we have never had this question from the experimenters):

```
node_id, authentication_type, hostname, port, username, iface_id1, mac1, ip1_1, netmask1_1
W2a, ssh-keys, n101-05.wall2.ilabt.iminds.be, 22, bvermeul,W2a:if0,0cc47a7b28a5,192.168.0.1,255.255.255.0,
W2b, ssh-keys, n101-03.wall2.ilabt.iminds.be, 22, bvermeul,W2b:if0,0cc47a7b28f3,192.168.0.2,255.255.255.0,
W1a, ssh-keys, n071-20a.wall1.ilabt.iminds.be, 22, bvermeul,W1a:if0,003048cfb565,192.168.1.1,255.255.255.0,
W1b, ssh-keys, n071-19b.wall1.ilabt.iminds.be, 22, bvermeul,W1b:if0,003048f02585,192.168.1.2,255.255.255.0,
```



- info about the image loaded on a node: this was indeed only possible through the manifest RSpec (XML). It's now implemented in jFed beta and will be there in the next release.

4.2.4 EXPERIMENT: PiAS (Televic)

Quote from report:

"We were happy with the functionalities as they are"

4.2.5 EXPERIMENT: Intelligent NFV Autoscaler (Modio)

Quote from report:

> We are overall satisfied by the offering we have had since it helped us in achieving our initial goal and objectives. To that end, the most important help for us was the technical support we received from our Patron, which made possible to overcome the technical issues we have had, in order to be able to reach to a demonstrator of our prototype during the 4th Fed4FIRE+ conference.

4.2.6 EXPERIMENT: Hammer (Univ. of Pireaus) (new since D2.9)

Quote from report:

"Installation of Openstack through ENOS at the Virtual Wall presents technical difficulties and despite the tutorial (outdated), we had several failures in our attempt to work with it."

Response:

- This is indeed a complex setup. We will improve this by doing a nightly test of the script and ENOS through the federation monitor platform.

4.2.7 EXPERIMENT: Elegant (Univ. of Coimbra) (new since D2.9)

Quote from report:

“Promote the use of configuration scripts instead of VM images”

Response:

- Yes, we try to do that as much as possible, but we’ll try to emphasize it even more “There is a lot of documentation, but somehow unstructured, specific to each site.”
- Yes, that’s true, but a choice from the federation to have autonomous documentation per testbed and maintained per testbed. We think this makes it easier to keep it up to date, but indeed has a difficulty for the experimenter. This should be tackled better in the SLICES infrastructure project by having more uniformity.
- Quote from report:
- Quote from report:

“Experiment deployments with error affecting only one node (and not crucial to the experiment): Instead of redeploying the full experiment provide the possibility to set the experiment as valid, even with a failing node”

Response:

- This has been made possible in jFed now

Quote from report:

“Enhance the intelligence in jFed:

- Provide links to the documentation regarding some functionalities that may not work (pcgen06 nodes do not support shared VLANs on vWall2, at the time of experiment)
- The multi command option should inform if testbeds use different connection settings (G5K requires SSH without proxy)
- Provide the means to request confirmation from other member on the disposal of resources. In some cases, experiments were deleted when others were working, some data loss was verified”

Response:

- Regarding the vlans for pcgen06 -> this was a bug in the testbed setup and will be fixed in 2022 with a refreshment of the testbed
- Different connection settings for G5K have been solved, so the proxy can now be used as well
- Disposal of resources: we have added some extra warnings in jFed before terminating an experiment

Quote from report:

"If possible, promote a "shared data storage" between testbeds:"

Response:

- Good point, in Fed4FIRE+ this is possible if you set it up yourself and share the storage over the internet, but you have to run an experiment for that. Permanently available storage over all testbeds is not available and difficult to implement. Smaller data sets/code can be shared through e.g. gitlab/github services. Storage between virtual wall and GPULab is shared e.g. In SLICES we will take this into account.

Quote from report:

"Promote the use of gitlab/github mechanisms to promote CI/CD operations in Fed4FIRE+:"

Response:

- We try to do so (especially internally at imec), but we'll try to promote it in Fed4FIRE+ as well

4.2.8 EXPERIMENT: KCCS (Umanis) (new since D2.9)

Quote from report:

"Discuss online between the Fed4FIRE+ experimenters about the progress of their projects and the problem encountered"

Response:

- That was the goal of the FEC meetings but due to COVID we could not organize these anymore and it is very difficult to organize networking online.

Quote from report:

"Add in the Virtual WALL testbed a Wattmeter to compute the power consumption"

Response:

- We have no such plans for the virtual wall. We have looked into this, and we do some monitoring from administration viewpoint, but there are a couple of problems: some of the nodes are twin nodes (2 servers on a single power supply) and from what we have learned from a couple of questions in the past, is that the time resolution of the info needed might vary from experiment to experiment. The solution is to look for a testbed that has power metering, e.g. NITOS now supports it and in the US also Cloudlab supports this (which is also usable with a Fed4FIRE+ account).

4.2.9 EXPERIMENT: BELA (lava.io) (new since D2.9)

Quote from report:

"A web-based JFed version would have been a very nice addition."

Response:

- For Fed4FIRE+ that is not foreseen, for SLICES it will be considered.

Quote from report:

"Although email has worked perfectly in our occasion, it is highly possible that a communication platform like slack for instant messages shared with all testers would have been very valuable."

Response:

- We have an email group forum (which is not used a lot). We see that most experimenters want to communicate directly with testbed and tool administrators rather than fellow experimenters.

Quote from report:

"The wiki page could include more common problems and troubleshooting."

Response:

- We try to solve the issues and prevent them for next users. We think that's a better approach. Also regarding trouble shooting, we put e.g. a feature in jFed to do link testing on all links in an experiment to be able to do quick testing if all links and interfaces are up and running.

Quote from report:

"Infrastructure monitoring tools per node and as a whole. Live updates directly in the platform. Would also be nice to see resource RAM/CPU usage for any of the nodes. And any errors that occur."

Response:

- For RAM/CPU monitoring, that can be done by the users (it varies per experiment what timescale or tools are needed, so very difficult to foresee something that is usable by everyone). We do not receive this question a lot.

Quote from report:

Google Cloud, Azure or AWS."

Response:

- That's a difficult one as those platforms are commercial while Fed4FIRE+ has free access policy.

4.2.10 EXPERIMENT: Ulysses (Hypertech) (new since D2.9)

Quote from report:

"In a few nodes, when the selected OS was Ubuntu 18.04 or later, the initial update and upgrade of the software components (the simple "sudoapt" commands) would take too much time to finish due to the fact that the nodes utilize ipv6 communication by default. Thus, even a single testing of the compatibility of a specific hardware/OS combination would sometimes take over an hour to complete, making the initial stages of the experimentation strenuous. To overcome this issue, we have enabled NATtedIPv4 Internet access. The hardware components were more than adequate."

Response:

- This is right. The Virtual Wall has by design public IPv6 addresses for all its nodes and private IPv4 addresses without NAT. It's a pity but IPv6 only is still not enough for all kinds of stuff. NAT can be enabled manually. This seems not straight forward for everyone as other cloud resources have direct internet access, always. We are currently redesigning the virtual wall internal networks so that IPv4 NAT will be enabled by default on all nodes.

Quote from report:

"One part of the experiment regarding an SDN-enabled switch, was not possible to be implemented since OVS did not work out of the box for us. Providing up to date images with preconfigured OVS and enhancing the existing tutorials is suggested for enabling seamless usage."

Response:

- Most experimenters using OVS deploy it themselves and do not use the preconfigured images. So they have indeed been outdated. We will rewrite the tutorial to start from installing OVS itself. This shows again that using images rather than install scripts is not a future proof way of experimenting if install scripts are possible. On the other hand, images are better for reproducibility, but here we see that this gets outdated as well…

4.2.11 EXPERIMENT: CloudBots (Canonical Robots) (new since D2.9)

Quote from report:

"jFed: Error log was not very descriptive"

Response:

- In the jFed GUI there is a tab 'errors' which gives short feedback to the user. The longer logs are meant for the jFed developers and through the jFed bugreport button this information is sent directly to the developers.

Quote from report:

"Virtual Wall: we could not use XEN VM nodes"

Response:

- That's true, they are currently not as stable as needed for experimenters. We advise to use the bare metal nodes to the users. We are currently rebuilding the testbed a bit (see also the native nat networking discussed above), and the XEN VMs should be more stable again.

4.3 Testbed W-ILAB.T (imec)

4.3.1 EXPERIMENT: MMT-IoT (Montimage)

Quote from report:

"Starting a project with a single reference to the documentation implies a steep learning curve for most of the researchers. In this sense, a presentation of a “project kick start” session would improve this situation, and make all the researchers aware of all the features Fed4FIRE+ has to offer"

Response:

- basic tutorials at the start are useful and necessary -> These people did not attend a tutorial at the start because of the continuous SME call (which is thus not aligned with FECs).
- a recorded basic tutorial can help as well, so we will work on an update of this.

Quote from report:

“More intensive support from the platform operators and a more clear and organized documentation. In this way, it would be possible to deploy and test more rapidly solutions and applications using the Fed4FIRE+ testbeds.”

Response:

- we continuously try to update and reorganize the documentation as good as possible
- a tutorial at the start helps also to answer starting questions

Quote from report:

“Valuable types of testbed infrastructures:

- Support for deploying commercial solutions.
- Integration of third-party external IoT networks in the Fed4FIRE+ infrastructure.
- Improved interoperability and APIs for simultaneously accessing and automating the access to multiple platforms.”

Response:

- The problem with commercial deployments is typically that they are not flexible enough for a testbed. We will try for 802.11ax to provide at least benchmarks of commercial equipment in the same room.

4.3.2 EXPERIMENT: Magic (Galgus)

Quote from report:

“One KVM for each type of device (different devices may require different integration steps).”

Response:

- Currently we have KVM (keyboard, video and mouse switch) for the nodes of type Zotac and DSS (same as MOBILE) and the only type which is missing is the APU type. We are looking to add a serial console for this.

Quote from report:

“A graphical monitoring tool to show the spectrum usage within the testbed in real time. This would be very useful to check the existence of interference, as well as to validate propagation models.”

Response:

- It is possible to monitor the spectrum in 2 ways. A very basic one is using “iwlist scan” which gives you the used frequencies. A more complicated one is using a USRP with GNUradio. This latter needs more setup time from the experimenter indeed. There is also a specific device (imec SE) but it’s a long time since someone used that one. We will update the documentation to reflect this.

Quote from report:

"The main bottleneck was the instability of mobile nodes. Nonetheless, we were able to minimize such bottleneck and finalize our experiments in time."

Response:

- This is true, the mobile nodes need a lot of support from testbed admins. These are mechanical parts of the testbed which are less stable than computing nodes.

4.3.3 EXPERIMENT: Simbed (InescTec)

Quote from report:

> It would be useful to have a new functionality allowing for radio spectrum reservation, as it happens for other resources such as nodes. Although difficult to enforce, this functionality would help experimenters announcing which channels are being used by them, so that the new experimenters can avoid that mutual interference. For example, NITOS has a spectrum analysis graph that allows checking which channels are currently being used, independently of the technology.
>
> Another thing that can be added is the possibility of changing the current node image without needing to switch for another experiment slice. In the case of w-iLab.t this was not possible, however NITOS offered the possibility to change a node image through OMF.

Response:

- Spectrum reservation: as said, reserving spectrum is difficult as the experimenters have full control and root access on all nodes. There are two solutions for this: reserve all resources so you are sure no one else is using parts of the spectrum, or monitor the spectrum. See previous experiment. One cannot really trust a gentlemen's agreement as also without intention (bug) something can go wrong. (on the other hand the lab is shared a lot of times for less critical experiments, where you monitor and use another channel.
- Changing the current node image:this is indeed something that is lacking in jFed/testbed API. It is natively possible on the testbed, but we need to look into adding this.

Quote from report:

> We would like to see some additional proactive support, especially in the beginning of the experiment. When the patrons are informed about the experiments that got approved for funding, they could organize a "bootstrap" skype meeting, giving a brief presentation on what they think would be the important tools/methodologies/known pitfalls/tutorials to follow in order to use their testbeds, having in mind the specific objectives of the experiment proposal.

Response:

- We try to scale this by giving tutorials, having good documentation, … most of the users get away with that (especially the open access users).

4.3.4 EXPERIMENT: Five (Feron)

Quote from report:

> In FERON we are mainly interested in radio access experimentation capabilities with both closed/commercial equipment (e.g. small cells, smartphones, dongles) and open prototype platforms (e.g. SDRs). This is undoubtedly a rapidly evolving area with new standards emerging for both traditional mobile services but lately also for vertical industries (for example automotive and IoT). In this respect, Fed4Fire+ support for these new communication technologies will be highly beneficial for early technology and related product testing. We report some possible additions in these directions:
>
> - 4G-LTE networks and devices supporting the latest Releases and UE categories. These will enable the testing of new capabilities, such as Gigabit LTE, and new application areas, such Cellular IoT (LTE-M/NB-IoT)
> - 5G-NR prototypes (when they become available)
> - LoRa/LoRaWAN for IoT experimentation, including development platforms such as Sierra Wireless mangOH, u-blox C030, Pycom, etc.
> - ITS-G5 development boards for IEEE 802.11p-based V2X experimentation
> - New high-end SDR boards like Ettus USRP N310 for supporting higher bandwidth applications (e.g. NR)
> - New small-form & low-cost SDR boards (e.g. LimeSDR/LimeSDR mini, bladeRF) for portable deployment and comparative performance evaluation with other solutions.

Response:

- Adding hardware is always challenging due to a cost involved and the fact that not all hardware is flexible enough to be put in a testbed and sometimes not useful enough to enough experimenters. That said, the w-iLab.t testbed has been extended with extra SDRs in the meantime, making the following SDRs available now: N210, B200/B210, X310, Zynq SDR. V2X experimentation should be available in Citylab/Smart highway testbed.

4.3.5 EXPERIMENT: SODA (Univ. of Montenegro)

Quote from report:

"We would have liked to have a RESTful API for experiment provisioning that we could use for automation purposes. The provisioning process in our case took quite a long time, over 20 minutes, which was not the case on IoT-lab testbed. "

Response:

- RESTful API: We have a jFed command line version (https://doc.ilabt.imec.be/jfed-documentation-5.9/otherjfedtools.html#experimenter-cli-2) which is typically used to automate experiments (together with the ESpec). We have worked on a RESTful API for the F-interop EU project, but there was no real demand to use this API outside the F-interop project. So it has not been maintained anymore. Apart from this, it should be perfectly possible to keep the experiment running, but to automate the experiments with the IoT sensors with a RESTful API (or with the opentestbed API) depending on the needed experimentation. This will also solve the 20 minutes delay (which is because a lot of software needs to be installed as far as we understood. Default setup time is about 2-5 minutes.

4.3.6 EXPERIMENT: Comfort-app (WINGS ICT Solutions)

Quote from report:

We think that the following would be really nice to have and in order to enhance the overall Fed4FIRE+ experimentation experience:

- A more user-friendly API for uploading/downloading files to/from the reserved nodes.
- A jFed tool/extension to save the image of a node in order for the experimenters to be able to reproduce the experiment in the future. One case where this functionality would be extremely useful is for example when you "Reboot OS" and the installed packages of the nodes are uninstalled. Then you have to set the nodes up from zero.

Response:

- More user-friendly API for uploading/downloading files: see above. When IPv6 is available at the client, standard scp tools can be used. If not, it is now possible to use web-based jupyter notebook to upload the data to the virtual wall and the copy it from there. It is not ideal. We constantly look how we can improve this. For some heavy users, we provide a VPN, which is not ideal either.

4.3.7 EXPERIMENT: Motive (Dilution) (new since D2.9)

Quote from report:

"The tutorial was outdated and the OMF and OML was no longer maintained."

Response:

- The tool ExpO (WP3) has been developed to substitute OMF and OML for easy orchestration during an experiment
- The w-iLab.t documentation was updated.

4.3.8 EXPERIMENT: Hammer (Univ. of Pireaus) (new since D2.9)

Quote from report:

"Sometimes the robot nodes were not able to dock back again, and we had to contact the patron."

Response:

- The robot nodes are physical moving hardware and indeed need a lot more support and maintenance. The manual support is indeed needed and is not available 24/7 (certainly with COVID also the presence at campus was a lot less).

4.3.9 EXPERIMENT: MMT-IoT (Montimage) (new since D2.9)

Quote from report:

"Transferring files sometimes slow (some bytes/s)"

Response:

- We were not aware of this, so difficult to debug. For sure, this should not happen. Connections are at least 1Gb/s on the wired connections.

Quote from report:

"JFed experimenter could be frozen if the manipulation is done too quickly, especially when requesting online resources (e.g. select the saved OS images)"

Response:

- We have seen this before, especially on linux. Typically jFed uses too much RAM. Several fixes have been done to avoid this. Lately we have not seen this problem anymore.

4.3.10 EXPERIMENT: SDR4IoT (rtone) (new since D2.9)

Quote from report:

"Availability of mobile nodes. Sometimes robot can't move. Smartphones and robots often have issues"

Response:

- This is the same as for another experiment: The robot nodes are physical moving hardware and indeed need a lot more support and maintenance. The manual support is indeed needed and is not available 24/7 (certainly with COVID also the presence at campus was a lot less).

4.3.11 EXPERIMENT: MANET4E (DSI) (new since D2.9)

Quote from report:

"From our perspective everything was okay and only one point could be improved. A shared datastorage (e.g.Dropbox or GoogleDrive) would facilitate the exchange of data between the testbed and third parties."

Response:

- This is the same as for another experiment, see above: Good point, in Fed4FIRE+ this is possible if you set it up yourself and share the storage over the internet, but you have to run an experiment for that. Permanently available storage over all testbeds is not available and difficult to implement. Smaller data sets/code can be shared through e.g. gitlab/github services. Storage between virtual wall and GPULab is shared e.g. In SLICES we will take this into account.

4.4 Testbed TENGU (imec)

4.4.1 EXPERIMENT: Scaling NewSum (SciFY)

Quote from report:

"The most difficult part during the experiments where to go through the hadoop and apache spark logs in order to perform debugging activities and troubleshoot errors. Installing commercial debugging tools could help optimize the development workflows and save time."

Response:

- this is of course at the experiment/application level. Commercial tools have a cost of course. Free tools are available but may have a learning curve. It's a difficult one. We typically look at user demand (if 10% of the users ask for this, then it becomes interesting to add this kind of tool, or if one user deploys it and takes the time to document this, we can add it to the documentation)

4.4.2 EXPERIMENT: DataTwin (Nissatech)

Quote from report:

"The existing offering is satisfactory. The only issue is that big data infrastructure should be updated regularly since new tools are appearing frequent"

Response:

- Because Tengu is more based on software stacks (than pure hardware infrastructure in other testbeds), it is more difficult to keep it up to speed with all latest stacks and new tools. This cannot be easily solved (needs a lot of manpower). People can always install newest tools themselves or use virtual wall, but it might need some time from them.

4.5 Testbed EDGENET (SU) (NEW SINCE D2.9)

Three main improvements have been implemented on EdgeNet:

- The control-plane components have been replicated on multiple geographic locations in order to increase the testbed reliability.
- Procedures have been prepared to prevent node saturation, using the node pressure eviction and pod overhead Kubernetes mechanisms.
- IPv6 support for EdgeNet is in preparation. The testbed is set to support IPv6 in the following months in production.

4.5.1 Experiment: neuropıl from pı-lar

Problems encountered / comments mentioned during the course of the experiment or listed in the report:

The experimenters appreciated the ease of use of EdgeNet. Since the testbed is based on pure Kubernetes, experimenters are not required to get familiar with another syntax to be productive with it. They also appreciated the ease with which they've been able to scale the experiments to multiple nodes.

The experimenters encountered the following issues:

- DNS resolution issues.
- Resource starving on the nodes, due to insufficient resources sharing mechanisms.
- Missing IPv6 support.

Actions taken / Modifications implemented by the testbed:

The EdgeNet testbed enabled a solution based on a virtual private network (VPN) to allow nodes behind NAT to participate in the cluster. In its current implementation, NAT-to-NAT communication are not yet possible. An issue caused the DNS servers of the testbed to be scheduled on node behind a NAT. As such, DNS resolution was impacted for a part of the cluster. This issue is addressed by constraining the DNS server deployment to the control plane nodes, which are reachable from all nodes, behind a NAT or not.

The testbed provides experimenters with the ability to quickly scale up and down experiments. This powerful feature comes with an overloading issue when many experiments run simultaneously. To address this issue, new node pressure eviction, and pod overhead configurations will be put in place in order to prevent experiments from saturating the nodes.

The testbed is not yet supporting IPv6. The team develops an Ansible playbook to configures the underlying infrastructure and enable IPv6 throughout the cluster. The EdgeNet testbed foresees providing dual-stack support, IPv4 and IPv6, in the second quarter of 2022.

4.6 Testbed NITOS (CERTH)

NITOS was upgraded with the deployment of 4 more USRPS B210 and 2 USRPs N310 towards the support of 5G experimentation, which gains more and more popularity lately. NITOS updated the images offered to the experimenters with the latest version of OAI, that includes additions made by CERTH. Moreover, NITOS continuously updates the local OpenSourceMANO installation, which is an NFV-MANO compliant orchestrator and provides NFV capabilities for the testbed.

New since D2.9: NITOS was upgraded with the deployment of LoRa testbeds in a city-wide environment, as well as in an agricultural environment. These testbeds were made available to the federation experimenters through efforts based on the 1st internal opencall. Additionally, a power consumption monitoring tool was developed in the context of the 1st internal opencall as a result of an opencall experiment feedback and a feature that the experimenter would like to have.

Some improvements / changes were implemented in response to specific comments made by some expermneters:

4.6.1 EXPERIMENT: F4F-LWA

Problems encountered / comments mentioned during the course of the experiment or listed in the report - Experimenter's comment:

"We were not able to do experiments only once, since there were no nodes with USRPs available (they were reserved from other experimenters). Other than that, the overall experience was very smooth."

Actions taken / Modifications implemented by the testbed

This experiment ran smooth, other than the problem with no resources during peak hours, there was no feedback regarding any major problem. Most of the problems and questions had been resolved through emails.

4.6.2 EXPERIMENT: FIVE

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No negative comments in the report. They mention that documentation was sufficient for their experiment.

The experimenters had prior experience with NITOS testbed and they could start their experiment from day 1 of their project.

Actions taken / Modifications implemented by the testbed

More than 60 e-mails were exchanged and 3 telcos were organized during Stage 1 and Stage 2, mostly to discuss the deployment and adaptation of a NITOS ITS software stack implementation.

4.6.3 EXPERIMENT: SIMBED

Problems encountered / comments mentioned during the course of the experiment or listed in the report- Experimenter's comments:

"For NITOS, we found important information missing from the website, such as the attenuators used and in which nodes they are installed."

"For NITOS it was necessary to use a nightly build to be able to properly access the testbed and make reservations."

Actions taken / Modifications implemented by the testbed

We have added the missing information in the documentation of NITOS (http://nitlab.inf.uth.gr/doc/wireless_example.html), so that new experimenters get informed about the attenuators currently attached to NITOS nodes.

IMEC has released a stable version of jFed, which fully supports NITOS testbed and provides the capability to reserve in advance resources.

4.6.4 EXPERIMENT: FED4QOE

Problems encountered / comments mentioned during the course of the experiment or listed in the report - Experimenter's comments:

- "In general, yes, however we have felt some hardware limitations to increase the number of UE/cell load in the LTE NITOS testbed."
- "In the NITOS testbed was not possible to test the impact of cell load on the performance metrics because the lack of UEs compliant with the UXPERT test requirements."
- "The EPC needs to be updated to be compliant with more recent LTE modems."

Actions taken / Modifications implemented by the testbed

- There are hardware limitations on the number of UEs that each femtocell supports (max 16/femtocell). This is defined by the hardware characteristics of the femtocells that we have adopted. The experimenters were suggested to move to experimental tools that may potentially provide more users/cell (e.g. OAI) but with less stability.
- We analyzed their experiment needs and suggested a workaround on this limitation by introducing more traffic flows per each UE. With this methodology, the measured indicators by the UXPERT tool would be the same as by having more UEs.
- The COTS NITOS EPC is supporting LTE Rel 10 (LTE Advanced) whereas the installed COTS femtocells are Rel.9 compatible. If more recent releases are needed, we provide all the necessary experimentation tools and infrastructure for running open source implementations of the LTE stack (e.g. OpenAirInterface/srsLTE) which support up to Rel. 14.

4.6.5 EXPERIMENT: SUNSET

Problems encountered / comments mentioned during the course of the experiment or listed in the report - Experimenter's comment:

"YES, we contacted some individuals from NITOS (CERTH) team to request assistance to provide missing components."

Actions taken / Modifications implemented by the testbed

Experimenter visited us in order to deploy his own equipment and run his experiment in NITOS. However, at the airport the batteries used for their IoT devices were confiscated. We provided the experimenter with batteries that we use in our own IoT devices, allowing this way the experiment to continue. No other comments or problems mentioned in their feedback.

4.6.6 EXPERIMENT: CDN-X-ALL

Problems encountered / comments mentioned during the course of the experiment or listed in the report - Experimenter's comments:

- Yes, the available resources are useful and inline what our expectations. However, we would like to comment that there are aspects related with USRP testbeds that might be useful to have. For instance, having access to dedicated USRP resources with dedicated LTE band RF duplexers/amplifiers/filters would help to improve the quality of some of our experiments (and stability). This could be indeed an option when reserving USRP resources where an experiment could also ask for specific LTE bands with specific RF frontend configuration.
- We decided to turn the number of connected UEs smaller to minimize the probability of having one of the UEs or elements from the eNodeB down suddenly. This aspect is quite related to our previous comment. We believe the reliability could be increased with dedicated LTE band RF isolation filters.
- The documentation is partially out of date (resources inventory and OMF syntax) and some atomic and updated examples should be published. It would be useful to include in the documentation the common/recurrent issues that may be found while experimenting and the instructions to solve them.
- Yes, the communication with the patron from NITOS testbed was continuous and fluent. We reported, connectivity issues, environment requirements, further documentation with examples, systems down and architecture requirements. We were quite satisfied NITOS patron provided support, fixing issues quickly and deploying new features to respond to our experiment demands very fast.
- Major aspects have been already addressed but we would like to emphasize that having access to dedicated USRP resources with dedicated LTE band RF duplexers/amplifiers/filters would help to improve the quality of the OAI SDR experience.

Actions taken / Modifications implemented by the testbed

- As the USRPs can be tuned in a wide range of frequencies (60Mhz - 6GHz), handling access for all the available spectrum will block almost all the other experiments. For this purpose, we provide spectrum monitoring tools for the frequencies in the testbed (through the NITOS scheduler) so as the experimenters tune their frequency usage. Specifically to the LTE experimentation, the frequency bands that can be used in the testbed are regulated by the wireless spectrum provider, so experimenters shall use only this spectrum.
- The experimenters run LTE in FDD mode, so installing such functionality would have made no difference at all. As they were running the LTE stack based on the OpenAirInterface implementation, we believe that their instability in the experiments were more related to using a "buggy" version of the software.
- The documentation has been updated accordingly.
- Continuous communication mainly through emails.
- Answered in comment 1 and 2.

4.6.7 Experiment: MECinFIRE (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Experimenter's comment: "Documentation was easy to follow and to setup an experiment. Perhaps the section on the availability of LTE dongles needs to be simplified."

Actions taken / Modifications implemented by the testbed

We have thoroughly documented the LTE experimentation procedures in NITOS by also adding videos with step-by-step tutorials (http://nitlab.inf.uth.gr/doc/lte.html).

4.6.8 Experiment: MECperf (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Experimenter's comment: "We implemented multiple experiments on the NITOS testbed. The preparation was rather complex. We had to configure the virtual machines for the different components that had to run in NITOS. The configuration required some effort as we had to install some additional software components (the MECPerf components and their dependencies, such as for example Java 11), and set up system configurations, for example to make a certain software component execute at startup. The process was not straightforward as the NITOS base images are based on quite old Ubuntu versions, such as 12 or 14, which are a bit different from the newest ones which we are used to. Also the automation of the experiments required some effort, as our experiments are made of multiple runs, and we had to implement some scripts to run all the commands on nodes in the right sequence, and to coordinate synchronization between nodes. As always happens when configuring and installing software, not everything runs smoothly at the first try, so the entire process required some time to be finalized."

Actions taken / Modifications implemented by the testbed

Based on the experimenter's comment we have updated the Linux images to also include the latest Ubuntu releases. Furthermore, we redesigned, and we are in the process of implementing the necessary changes in the NITOS software responsible for image provisioning, in order to allow future experimenters to create their custom images more easily.

4.6.9 Experiment: EXPOLRA (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Experimenter's comment: "The documentation that is available online can be improved, by adding all the tutorial information that was offline provided to us by the NITOS testbed team."

Actions taken / Modifications implemented by the testbed

Documentation about LoRa capabilities of NITOS and instructions on how to use the testbed has been added in NITOS documentation.

4.6.10 Experiment: IoTaaS (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No problems/comments by the experimenter in the report.

Actions taken / Modifications implemented by the testbed

Typical support by exchanging emails was enough for the experimenter.

4.6.11 Experiment: PAWL (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No problems/comments by the experimenter in the report.

Actions taken / Modifications implemented by the testbed

The experimenter during its Stage-2 experiments, visited NITOS premises and integrated his own equipment in NITOS testbed for the purposes of his experiments. Thus, all the required support was given on-site, and the experimenter was also given support on how to operate the oscilloscope in NITOS premises for part of his experiments.

4.6.12 Experiment: Precomind (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No problems/comments by the experimenter in the report.

Actions taken / Modifications implemented by the testbed

The experimenter proposed a power consumption monitoring tool as a feature that he would like to have in Fed4FIRE+. Based on his feedback a power consumption monitoring tool was added to NITOS testbed through the internal on-demand extensions OpenCall.

4.7 Testbed IOT LAB (MANDAT)

During the period covered by this report, IoT Lab has technically joined the Fed4FIRE+ federation of testbeds. Thanks to the strong collaboration between the engineers of Mandat International and imec, the integration was mainly planned during the year 2019 taking into account all the technical constraints discovered in the initial phase of technical study done in 2018. Concretely, the IoT Lab integration was effective since the 5th February 2019: indeed, since this date, the status of the IoT Lab testbed is monitored online on Fedmon: https://fedmon.fed4fire.eu/testbed/iotlab. Afterwards, Mandat International has done some improvements and updates on the IoT Lab side to correct some bugs and ensure a better service availability at the long run.

Since D2.9, the connection between the IoT Lab testbed and the central Fed4FIRE+ server hosted and managed by IMEC was improved to ensure a better reliability and scalability. The uptime is 99.46% for the last year period; it corresponds to a downtime of 47.68 hours. The downtime is essential due to upgrades, updates and renewals of certificates. In parallel, an update of the F-Interop platform was done to ensure the access to the F-Interop platform through Fed4FIRE+, in particular through the jFed tool.

4.8 Testbed NETMODE (NTUA)

NETMODE testbed aims to shift from a wireless testbed to a modern testbed for Mobile Edge Computing experimentation. Towards these directions, an educational robot, i.e. Waveshare AlphaBot, and two Raspberry Pi 3 devices are added in the testbeds.

Since D2.9, NETMODE testbed continuously evolves from a wireless testbed to a modern testbed for Edge Computing experimentation. Granted by the internal open call for testbed extension, we added two mobile ROS-enabled robots (Turtlebot 3 Burger) for Industry 4.0 experimentation, various sensing modules (cameras and LiDAR), wireless nodes for the outdoor testbed and wireless nodes with augmented capabilities for indoor experiments.

4.8.1 Experiment: ULYSSES (F4Fp-SME-COD201103) (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

According to the feedback of the experimenter (HYPERTECH), the following comments about NETMODE testbed are provided:

- For the specific experiment, the specific version of some software tools was necessary, and they had to install it manually.
- For easier reproduction of the experiment, it would be useful to store locally container images with pre-installed applications, which were used in the experiment.

Actions taken / Modifications implemented by the testbed

In order to improve our testbed and considering the previous comments, the following improvements are done/scheduled:

- Regarding the first comment, it is not possible to pre-install experiment-specific software tools. This can be done only on-demand.
- Regarding the second comment, the storage of experiment-specific images is not possible in NETMODE or any other Fed4FIRE+ testbed because this requires large extra storage, which is not available at this moment.

4.9 Testbed GRID5000 (inria) (NEW SINCE D2.9)

The main improvement is the completion of the support for the Fed4FIRE+ API. Other tools were implemented in the context of T3.2.

4.9.1 Experiment: ELEGANT

Problems encountered / comments mentioned during the course of the experiment or listed in the report

From the report: "The user experience was excellent in terms of support, and in terms of accessibility from the testbeds responsible. Indeed, the requests made to Grid5000 or to vWall2 had always a prompt reply."

This experiment was among the first to use stitched links between Grid5000 and VirtualWall. It was very useful to have feedback from real users on this feature.

Actions taken / Modifications implemented by the testbed

None needed (outside normal experimenter support during the project).

4.9.2 Experiment: FRIDA Stage 2

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Problems while setting up stitched links between Grid5000 and VirtualWall. This was due to a configuration problem on the GEANT side, which was solved after completion of the project. (It was not a blocking issue for this project.)

From the report : "In summary, from Tengu we got a new perspective about the technology evolution, to pursuit a Kubernates approach. From GPULab, the convenience of using Jupyter Notebooks and from Grid5000 the hardware and resources to test everything."

Actions taken / Modifications implemented by the testbed

In response to this feedback (and to other feedback) we pursued the work on adding Jupyter notebooks support to Grid5000 (described in T3.2).

4.9.3 Experiment: GoldenOWL 2.0

Problems encountered / comments mentioned during the course of the experiment or listed in the report

From the report: "Fed4FIRE+ offering was perfectly matching our needs (in particular through the Grid'5000 testbed), allowing us to implement quickly and in a (relatively) low-risk way the changes we thought were needed before launching to market."

Actions taken / Modifications implemented by the testbed

None needed (outside normal experimenter support during the project).

4.9.4 Experiment: DENOW

Problems encountered / comments mentioned during the course of the experiment or listed in the report

From the report: "We were missing access to high computation infrastructure where the model training could be executed, and the large input data stored. Testbed GRID5000 was an excellent match for this purpose."

From the report: "The technical offering was up to date with what we know from experience in our previous projects on other platforms. The positive aspect is how transparent and organized everything was on GRID5000 (job monitoring dashboard) and ability to forecast when is the busy time on the nodes. We have no negative aspects or requirements that were not able to be fulfilled."

From the report: "Before the experiment started, we had a video call with testbed Patron. This meeting was to introduce the experiment on a more detailed level, but also to get the instruction of how to use the GRID5000.

The preferred way of the Patron was to guide us towards the existing online documentation and Wiki pages, which proved perfectly throughout our experiment."

Actions taken / Modifications implemented by the testbed

None needed (outside normal experimenter support during the project).

4.9.5 Experiment: DRX

Problems encountered / comments mentioned during the course of the experiment or listed in the report

From the report: "Lucas and his colleagues at INRIA and GRID5000 always and instantly helped us when we needed help. It's great to work closely with technical cloud experts."

Actions taken / Modifications implemented by the testbed

None needed (outside normal experimenter support during the project).

4.9.6 Experiment: FIONA4DOCKER

Problems encountered / comments mentioned during the course of the experiment or listed in the report

From the report: "We used it to manage resources and jobs in Grid5000. It was simple and effective. We used it to reserve specific nodes that we wanted, more than 1 node at the same time, and check their details."

From the report: "We have some problems when reserving specific nodes in Sophia. A lot of times it got stuck and did not reserve our nodes."

Actions taken / Modifications implemented by the testbed

We use the Sophia site mentioned by the experimenters to deploy new developments in a real environment before deploying them to other sites. As a result, it is often unstable. We clarified its status in documentation so that users avoid it and the lower reliability is a problem for them.

4.10 Testbed I2CAT OFELIA (i2cat)

During period 2, and certainly the beginning of 2019, we implemented the logic to serve a privacy section in the OFELIA i2CAT Control Framework stack (accessed via the https://f4f.lab.i2cat.net/privacy/ endpoint). This allows new users to review and accept (or not) the OFELIA i2CAT Terms & Conditions in a fully integrated form with jFed, or to alternatively point their browsers to read the conditions. Their Fed4FIRE certificate is used to extract information and save their preferences on the above stated.

Since D2.9, a number of improvements were made on the testbed.

Two of them relate to development. Specifically, regarding the (i) development of a monitoring system for the establishment of the Service Level Agreement (SLA) on each experiment; and (ii) the development and refactor of the mechanisms to obtain, persist temporarily and validate the user's consent on the processing of their data upon the user's consent decision. These required integration with the SLA & reputation systems and with the jFed built-in browser and are described in detail in D3.4 and D3.6, respectively.

Besides the development works, there were a number of operational efforts, indicated as follows. First, some troubleshooting and fixes took place for the network access – mostly on the SSH access to the nodes provisioned on the testbed. To that end, a layer-3 GRE tunnel was established between imec and i2CAT so that the Fed4FIRE+'s jFed tool and specific indications were defined on the firewalls to establish direct connection to the nodes via SSH. Besides this, there was some work on the update and/or revocation of trusted X509 certificates (named "trusted roots") on the testbed Aggregate Managers (AMs).

4.11 Testbed SMARTSANTANDER (UC) (new since D2.9)

During this reporting period we have implemented new notification protocols as a new functionality of the IoT API. Specifically, websockets and mqtt have been implemented. Besides, we have extended the previously existing LoRaWAN infrastructure coverage area by deploying new gateways on selected locations. Lastly, new LoRaWAN-based parking sensor devices have been deployed into the SmartSantander platform.

4.11.1 Experiment: SECTOR/Smart-IoT

Problems encountered / comments mentioned during the course of the experiment or listed in the report

- Minor comments mentioned in the report, all interactions with testbed owners were positive and helpful and the resources were offered without problems after an initial issue with mobile IoT nodes.

  "For resources we used to access to SmartSantander, the availability was overall very good. In the beginning, several mobile sensors were showing older values, but it got very quickly corrected through Patron communication-channel."

  "The communication with support (in our case testbed Patron) was outstanding. We communicated with emails, skype calls and a chat tool (Teams) which integrates the development repository. In order to resolve any questions and problems of technical nature, these were efficient usually to find a solution within one day the longest."

- Expectations on how the information was going to be received did not match the real scenario, but it was easily overcome. Besides, experimenter subscribed to data streams from all temperature sensors, without filtering it by temperature range. As a result, malfunctioning ones providing out of range measurements were also part of the experiment. However, they managed to filter that information and provided feedback about it.

  "Running the experiment, we experienced a few challenges in the beginning. The API-endpoint provided by SmartSantander testbed was not providing the temperature-sensor measurements-data on a regular time interval and we needed to adjust the importing of it into a database. Another issue was that some sensors reported values outside of any reasonable range. To overcome this, we have added a check to the data import and created a report for the testbed with the faulty sensors. Experiment execution has for some points exceeded our expectations. Initially, we only planned to include data from the stationary temperature-sensors. While iterating few initial challenges, the testbed has included the mobile-sensor data as well in real-time to the API-endpoint. This has resulted in improved measurements for our web-app and a more granular Urban Heat Island map visualization"

Actions taken / Modifications implemented by the testbed

We needed to fix some temporary problems, but no modification to the testbed infrastructure was carried out. At some point, the experimenter asked about the possibility to support websockets or mqtt technologies to enable asynchronous notifications without the need to set-up a listening server on the client side (as it happens with REST Hooks). While it wasn't feasible for the duration of the experiment, we decided to implement the functionality during the reporting period.

4.11.2 Experiment: Fed4cities (STAGE 1 and 2)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No problems or comments mentioned in the report, all interactions with testbed owners were positive and helpful and the resources were offered without problems

“To deal with only one service provider indeed facilitates the use of the testbed. We also managed to interact individually with each resource provider without any problem.”

“We used three testbeds and more than 350 resources among them. All worked out nicely and we were able to implement the interoperability when it was needed.”

Actions taken / Modifications implemented by the testbed

No actions were taken as experimenter did not request any functionality not yet available nor had any problem accessing SmartSantander sensor information.

Nevertheless, as one of the main applications of the stage 2 experiment was related to smart-parking domain, we have continued deploying new parking sensor infrastructure in the city of Santander. In particular, we have enhanced our LoRaWAN network coverage to explore the capabilities offered by this technology when applied to smart-parking solutions.

4.12 Testbed TRIANGLE (PerformLTE) (UMA)

Implementation of new measurement points adapted to the experiment requirements. These measurement points helped to evaluate the main aspects target of analysis also increasing the accuracy of the results reported previously.

4.12.1 Experiment: BOOST (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

The execution of this experiment did not present any important issue. As a note, considering the situation and restrictions during the timeframe of this experiment it was decided early on to perform it remotely, providing VPN access to the experimenters and sending to the testbed via postal services the required devices. This resulted in a small delay in the reception of the devices, which was outside the control of both the experimenter and testbed, however this was rapidly corrected once the devices where installed.

Actions taken / Modifications implemented by the testbed

For the execution of this experiment only minor modifications needed to be performed in the Testbed. In order to facilitate the remote access to the equipment for the experimenter a separate computer was installed and made accessible via VPN and Remote Desktop Connection, and that had a direct connection to both the Power Analyzer and the BOOST IoT device.

4.12.2 Experiment: OTTVPN (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No important issues occurred during the execution of this experiment, however, after the initial planning stage the phone reserved for this experiment broke, which resulted in a modification of the plans: The initial idea was to perform this experiment using an emulated environment provided by the 5G UXM, but due to the impossibility of obtaining a connectorized device during the timeframe of the experiment it was decided to use the real 5G network deployment.

Actions taken / Modifications implemented by the testbed

As in the case of BOOST, all actions were performed remotely by the experimenter by using VPN access and a dedicated computer where all the required devices were connected.

This experiment made use of the NEMO Outdoor tool in order to obtain real time measurements from the phone. Since the logs generated by this tool needs to be processed by a separate application that was not available to the experimenters it was decided that, after the execution of each trial the testbed personnel would take the generated results, process them, and send readable CSV flies to the experimenters in the next day.

4.13 Testbed LOG-A-TEC (JSI)

During the reporting period several new software solutions and updates were performed:

- Contiki-NG operating system was ported in order to support TSCH standard for IIoT applications
- Enabling Continuous Integration (CI) approach
- Implementation of the Zero-Touch configuration WiFi solution

4.13.1 EXPERIMENT: MMT-IOT – Stage 2 (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

To conduct the experiment, devices provided by Montimage were included in the testbed. During the experiment, some MMT-IOT devices malfunctioned, requiring a firmware update. During the experimentation period, two of the Log-a-Tec nodes failed and were not used for the experiments. Since the Log-a-Tec testbed does not provide straight forward access to the testbed nodes, some limitations arose when conducting more comprehensive experiments with different configurations. It turned out that a simple overview of the experiment network would improve usability and speed up the execution of the experiments.

Actions taken / Modifications implemented by the testbed

To perform firmware updates on additional devices in the testbed, remote access via a VPN was set up for the remote debugging, flashing, verification of application execution, and obtaining the experiment results in an easier manner.

Additional analysis was performed on malfunctioning nodes. The flaw in the infrastructure board design was identified. In some rare cases, the LDO of the power supply oscillated, preventing the devices from turning on properly. The problem was solved by replacing a single capacitor on the board.

During the experiments, the administrators of the LOG -a- TEC testbed were in constant contact with the Montimage team. By changing the device firmware and configuring the experiment application, different experiment scenarios were run to handle most of the planned activities.

To improve the user experience during experiment execution an upgrade of the testbed management system with the Experiment Control and Monitoring System (EC &MS) service was performed as part of an internal open call.

4.13.2 EXPERIMENT: LOWINTER – Stage 1 (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Although the experiment went relatively smoothly, a few minor problems occurred. First, the initial test application was designed to interfere with communication between ELMIBIT LoRa- WAN devices. However, creating noise on only one channel in the 868 MHz band was unsuccessful - the devices communicated with no lost packets. Second, during the execution of the experiment, there was no feedback on whether the devices received and successfully launched the experiment application, as the testbed nodes only generated noise and there was no additional communication with them that could have confirmed correct execution.

Actions taken / Modifications implemented by the testbed

To generate noise on different channels simultaneously, we updated the process of deploying experiments in a testbed, allowing more complex applications to run - each node was given its own application. This allowed the nodes to generate noise on the entire 868 MHz spectrum and successfully jam the ELMIBIT devices.

To address the second issue, an SSH connection was made to each device during the experiment to verify that the application was running correctly. An update of the testbed management system, in which we implement EC &MS, enables real-time observation of the experiments and confirmation of their execution. All devices automatically inform the management server about the execution status of their application.

4.13.3 EXPERIMENT: WIBRO – Stage 1 (new since D2.9)

Problems encountered / comments mentioned during the course of the experiment or listed in the report

Since the experiments were conducted in a remote industrial environment using mobile nodes, the experiment environment was not known and several issues regarding power supply for the nodes and Internet connectivity had to be addressed.

In addition, this was the first experiment where the WiFi interface integrated into the infrastructure part of the Log-a-Tec node was used for the experiments.

Actions taken / Modifications implemented by the testbed

To cope with the deployment of the devices in a new, unknown environment, we have developed an ad-hoc technique where a node acts as a provisioner and offers WiFi AP to which other devices automatically connect. These solutions allow the nodes to be set up in the new test environment more quickly and the experiments to be performed faster. In addition, the devices were adapted to run on battery power.

To handle the WiFi experiments, the testbed administrators prepared a series of tests and experimental applications that explored WiFi connectivity in the factory environment and were later deployed in the field.

4.14 Testbed IRIS (TCD) (new since D2.9)

There have been many minor improvements in the testbed since 2020 including bug fixes etc, but in this section we are going to focus on the more substantial upgrades to the testbed.

The Iris (OpenIreland) testbed has undergone some very serious upgrades since January 2020. These upgrades have been supported by funding from Science Foundation Ireland. The Iris (OpenIreland) reconfigurable 5G and beyond radio, optical, and cloud testbed headquartered at Trinity College Dublin, provides virtualized 5G radio, optical transmission equipment, software virtualisation, Cloud-RAN, Network Functions Virtualisation (NFV), and Software Defined Networking (SDN) technologies (including OpenFlow and Netconf) to support the experimental investigation of the interplay between future networks and new radio. The testbed has been upgraded to include indoor and outdoor 5G new radio, cloud, and optical transmission equipment deployed within Trinity College Dublin, around the Dublin Docklands area, and out to the DCU Campus in North Dublin. Our facility pairs underlying flexible radio, reconfigurable optical access networks and computation resources with various hypervisors in the form of software defined radio (SDR) frameworks, virtualized network functions (VNFs), and SDN network slicing capabilities, running across a multi-tenant and multi-cloud OpenStack environments to realize various reconfigurable research and experimentation configurations. The testbed is principally based on open-source software including Openstack, Open Source MANO (OSM), Goldstone, OpenAirInterface 4G/5G and srsRAN (4G/5G) which are implemented over open interfaces (i.e., O-RAN, ODTN). Iris (OpenIreland) is ideally equipped to support research towards the combination of SDR and radio slicing, SDN, network functions virtualisation, Quantum communication, and physical layer approaches into coexisting and coherent next-generation commercial networks, including but not limited to 5G.

The optical capabilities of the Iris (OpenIreland) testbed include over 1,700 km of fibre spools, ROADMs, amplifiers, coherent transponders, optical signal monitors, in addition to typical optical laboratory equipment, such as spectrum analyser, real-time scope, optical filters, etc. The testbed is also fully reconfigurable supported by a large port count optical fibre switch, which manages the topology for any experiment. This equipment has been integrated into the testbed in 2021.

The Radio resources include National Instruments (NI) High Performance Software Defined Radio USRPS (see Figure 2): 24 NI N210 USRPs, 12 NI USRP X310s, 14 NI B210 USRPs, 4 NI B210 minis, and 2 NI USRP N310, supporting DC to 6 GHz frequencies and up to 100 MHz of baseband bandwidth. All USRPs are connected to the optical and cloud components of the testbed network via optical cables. To expose the functionality of USRP equipment for applications, Iris (OpenIreland) employs a variety of open-source radio hypervisors that freely enable prototyping of wireless systems, as exemplified by GNURadio, srsRAN 4G/5G, and Open-Air Interface 4G/5G. The abstract representation of the different layers of the testbed including Functional Elements (Hardware), Virtualisation Layer, etc., are represented in the figure below.

Since mid-2020 the Iris testbed team has been testing and deploying the open source OpenAirInterface 5G non-standalone (NSA) framework. We have been deploying OpenAirInterface 5G Standalone (SA) since June 2021. Both OAI 5G NSA and SA modes have been available to Fed4FIRE+ experimenters since August 2021. These toolkits are experimental and improving daily. As it stands today (December 2021), the Iris testbed team can get NSA and SA mode transferring data at 87Mbps to commercial UE devices over the air. We also offer a OpenAirInterface SA simulation toolkit to experimenters. srsRAN NSA has been available to experimenters at the Iris testbed (Oct 2021). srsRAN also has a NR UE which is available to deploy at the Iris testbed. Open Source 5G core solutions supporting these frametwork include: OAI 5G Core Network supporting NSA, and SA modes5, Open5GS6, and srsRAN EPC7. We also have a number of UEs to support NSA and SA testing including: OPPO Reno 5G, CPH1921 (5G bands 78 NSA/Sub6), Xiaomi Mi 10 Global Version 6.67, and 3 x Quectel RM500Q-GL 5G Modems. Remote accessibility to these nodes is supported by the scrcpy (v1.21) framework, which provides display and control of 5G NR Android devices, and Quectel UE devices connected via USB to server or laptops.

Optical and Wireless equipment are connected to a private computational cloud (see figure) orchestrated primarily by the Canonical suite of technologies including MAAS (Metal as a Service), containers (Docker, LXD, LXC, Kubernetes), cloud platform (OpenStack and OpenStack Microstack), and operating systems (Ubuntu), enabling Iris to easily build and deploy multiple isolated testbed environments supporting dynamic virtualised experiments.

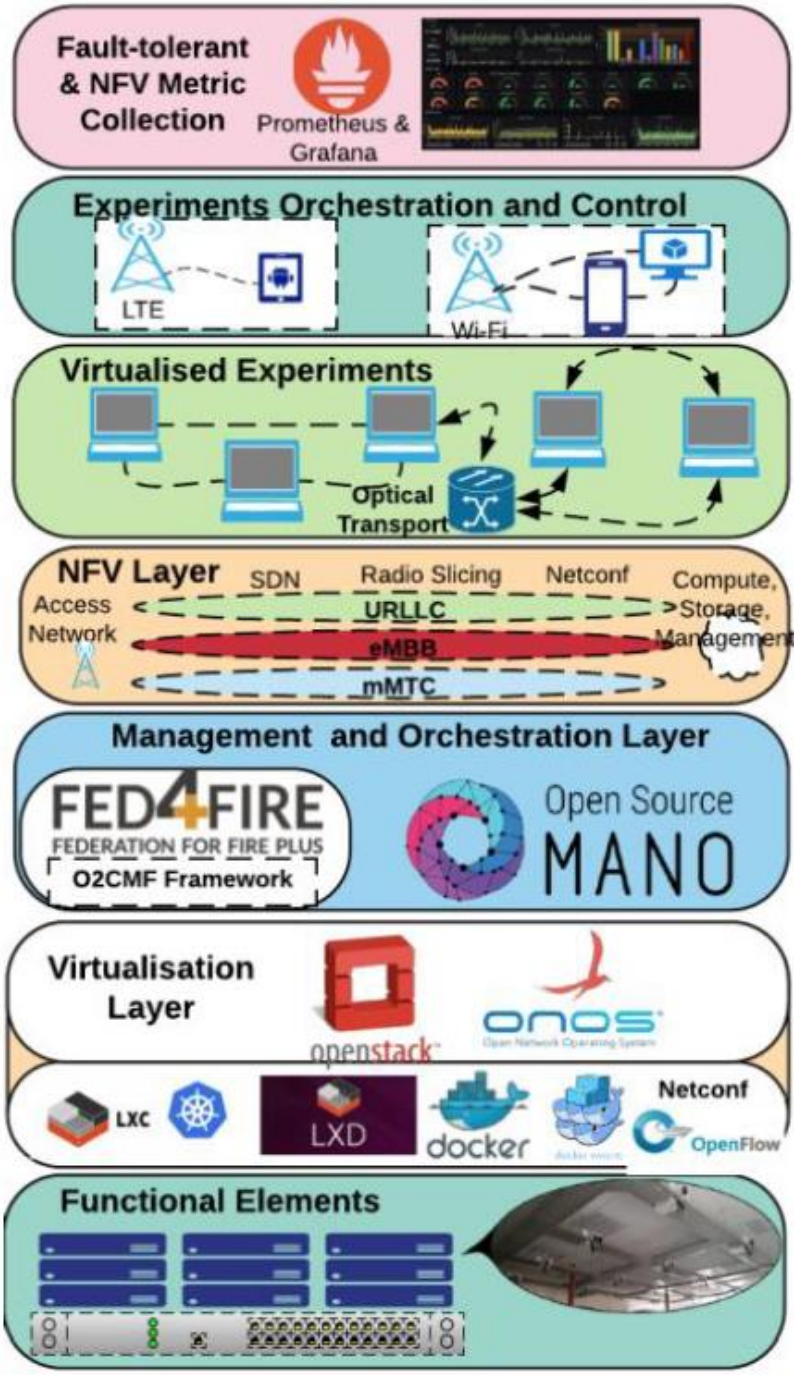


Figure 11: Open Ireland Testbed Architecture

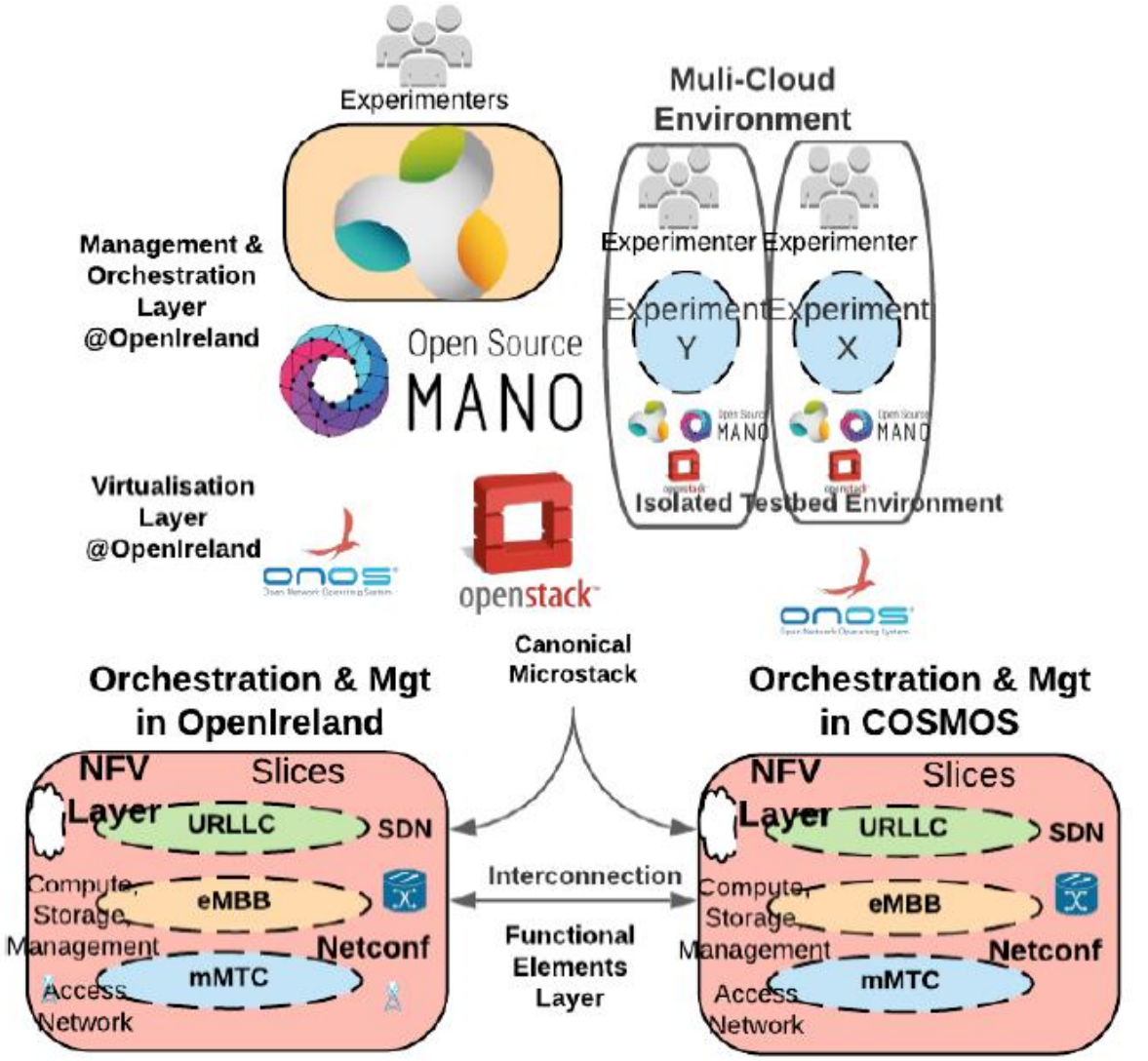


Figure 12: Open Ireland virtualised experiments

Iris (OpenIreland) uses a multi-cloud environment based on the Canonical suite of technologies to support experiment isolation – see Figure 12. The installed Virtualized Infrastructure Manager (VIM) utilised for each experiment is Canonical Microstack.OpenStack, which is a minimal version of OpenStack with only the core components installed. The objective of this architecture is to eliminate interference between experimentation units in randomized experiments across testbed equipment. This type of architecture effectively provides experimenters with their own dedicated testbed. This is supported and experimentation is simplified further using OSM. OpenStack is one of the main VIMs supported by OSM.

In parallel to all of these activities we are upgrading the testbed to support environmental sustainability. Towards this goal, we have installed 9 x Raritan PDU outlets (Raritan PX3-5190CR x 3, 48DCWC-16-2X100-A0 x 5, WC- PX3-5498V x 1) to remotely manage power to servers and USRP devices. The objective is to turn on and off servers based on on-demand. It is also possible for experimenters to remotely monitor power consumption of the servers and USRPs PDU outlets, which can support researchers with energy innovations. Currently we have built an API to remotely control and monitor the PDU devices. The remaining objectives include integration with jFed, and the development of a usage prediction tool.

Finally, the Iris team are also continually evolving tutorials for experimenters. Towards this goal, one of the recent tutorials added (Dec 2021) involves integration with the Pervasive Nation LoRaWAN IoT testbed. This tutorial supports experimenters using Seeeduino LoRAWAN boards at the Iris testbed, which is supported by the ThingsNetwork and the CONNECT Centre Pervasive Nation LoRaWAN IoT testbed.

4.14.1 EXPERIMENT: AERO 5G - Stage 1.

Problems encountered / comments mentioned during the course of the experiment or listed in the report

The AERO 5G had some minor issues related to bugs in the reservation system, which were resolved quickly, and a number of minor challenges related to using hardware and software components in the testbed. The Iris team also supported the experimenters getting on site access to the Iris testbed during the Stage 1 experiments, as well as by providing technical assistance when configuring and provisioning the srsLTE eNodeB, EPC equipment, and sysmocom SIM cards.

Actions taken / Modifications implemented by the testbed

The on-site access to Iris was required as experimenters were using AR technologies on Android Handsets. Towards supporting remote access and testing using Android handsets via the jFed framework, we now utilise the scrcpy framework. The scrcpy application provides display and control of Android devices connected via USB to servers or laptops. At Iris we virtualise the USB interface using OpenStack on Servers, which enables VM access to the Android device attached to the USB interface. The GUI can be invoked remotely via command line. Additionally, we can also support GUI access to Android devices using laptops and TeamViewer. These upgrade have proven useful for other experiments accessing 5G handsets for experimentation.

4.14.2 EXPERIMENT: VCOM – Stage 1

Problems encountered / comments mentioned during the course of the experiment or listed in the report

ALLBESMART were seeking the possibility of testing 3GPP 5G-NR (NSA and SA) technologies on Fed4FIRE+. They would like a dedicated service to set up the experiment to speed up the implementation.

Actions taken / Modifications implemented by the testbed

When the VCOM experiment was executed in 2020, 3GPP 5G-NR (NSA and SA) technologies were only commercially available, while open-source alternatives were in their infancy. Commercial technologies prove difficult to support experiments like VCOM, as experimenters need to make modifications to the source code. Based on our experience access to source code is not available from commercial solutions.

Since mid-2020 the Iris testbed team has been testing and deploying the open source OpenAirInterface 5G non-standalone (NSA) framework. Since June 2021, we have been deploying and testing the OpenAirInterface 5G Standalone (SA). Since August 2021, both OAI 5G NSA and SA modes have been available to Fed4FIRE+ experimenters. These toolkits are experimental and improving daily. As it stands today, the Iris testbed team can get NSA and SA mode transferring data at 87Mbps to commercial UE devices over the air.

Since October 2021, srsRAN NSA has been available to experimenters at the Iris testbed. srsRAN also has a NR UE which is available at the Iris testbed.

Open Source 5G core solutions available at the Iris testbed include: OAI 5G Core Network supporting NSA, and SA modes , Open5GS , and srsRAN EPC .

Iris also has a number of UEs to support NSA and SA testing including: OPPO Reno 5G, CPH1921 (5G bands 78 NSA/Sub6), Xiaomi Mi 10 Global Version 6.67, and 3 x Quectel RM500Q-GL 5G Modems.

These toolsets also support frameworks such as scrcpy (v1.21), which provides display and control of 5G NR Android devices, and Quectel UE devices connected via USB to server or laptops.

4.14.3 EXPERIMENT: 5G-VTWIN – Stage 1

Problems encountered / comments mentioned during the course of the experiment or listed in the report

No IP cameras available at the Iris testbed.

The performance of the VPN in terms of networking latency did not guarantee a good connection to process video streams in real time manner at the IMEC GPU lab testbed. Moreover, the only available GPU at IMEC testbed was a NVIDIA GeForce GTX 950 which was underperformance with respect to the one provided by IRIS, which was a NVIDIA GeForce GTX 1060 GPU.

Actions taken / Modifications implemented by the testbed

The Iris testbed team installed two D-Link 4703e high-sensitivity 3-megapixel progressive scan CMOS sensor cameras, which were placed overlooking the street below the building (Fenian Street, Dublin 2, Ireland). These or similar devices can be installed for experimenters to support ML training activities. This was the original requirement set out in the 5G-VTWIN proposal for the Iris testbed team.

Problem: The goal of the 5G-VTWIN experiment was to classify road users in real-time including: pedestrians, cyclists, cars, trucks and buses. However, the VPN Link to IMEC GPULab provided inadequate bandwidth to support real time HD quality data streams. Additionally, the GPU (NVIDIA GeForce GTX 950) available at GPULab was not very powerful.

Solution: The Iris team offered to set up the 5G-VTWIN team with a NVIDIA GeForce GTX 1060 GPU. This enabled the 5G-VTWIN experiment to complete the experiment and submit a report.

Since the completion of the 5G-VTWIN project, the Iris testbed team have made two GPU units available to Fed4FIRE+ experimenters. These include the NVIDIA GeForce GTX 1060 GPU used by the 5G-VTWIN experiment, and an Alienware NVIDIA GeForce GTX 1070 GPU laptop. This equipment has already been used in a subsequent Fed4FIRE+ experiment. Furthermore, we are also planning to purchase further GPU equipment to support internal TCD and external Fed4FIRE+ experimenters.

5 DOCKER-AM AS EXAMPLE AM

This section describes the features and install instructions for the Docker AM (supporting IPv6 resources as well!), that we advise as example/template of an Aggregate Manager. Code for this can be found at https://github.com/open-multinet/docker-am.

This was updated since D2.4 in the section about trusting root certificates and specifically also towards the new portal.

5.1 SUPPORTED AGGREGATE MANAGER FEATURES

- Every basic feature (Allocate, Provision, Delete, Status, ListResources, Describe, Renew)
- Some PerformOperationalAction call are supported
  - geni_update_users : Update SSH authorized keys or add a user
  - geni_reload : If you want to "reset" your container
  - Other options have no effect
- You can provide a sliver-type to get different kind of containers (for example limited memory or CPU container). Check the advertisement RSpec, and have a look at gcf_to_docker.py for details.
- Install a custom docker image by providing a name from a DockerHub or a URL to a Dockerfile or a ZipFile containing a Dockerfile and dependencies.
- Restart the AM without losing the state of existing slivers: Running docker containers will keep running when the AM stops, and can be controlled again when the AM restarts. (You can safely remove am-state-v1.dat to clear the state and thus force config reload. You will need to kill any running docker containers manually in that case.)
- Multiple physical host for Docker. That means you can increase the scalability easily by setting up a new "DockerMaster" on remote host. To scale the setup, integration with kubernetes is probably preferable.
- install and execute can be used to install a zipfile in a specific directory and execute commands automatically when the container is ready.
- IPv6 per container can be configured in addition to the IPv4 port forwarding of the host.
- The is demo code that can be used as a basis to customize the AM. Two features are demonstrated in this code: ** Supporting custom non-container external resources. (See resourceexample.py) ** Automatically adding a gateway proxy per slice. (See "proxy" in the configuration parsing)

5.2 HOW TO INSTALL THE AM?

5.2.1 Dependencies

apt-get install -y python2.7 python-lxml git python-m2crypto python-dateutil python-openssl libxmlsec1 xmlsec1 libxmlsec1-openssl libxmlsec1-dev python-pip

- Pyro4 is also required (for remote dockermasters):
  pip install pyro4

5.2.2 Download source code

- First clone the git repository
  git clone https://github.com/open-multinet/docker-am.git
- Initialize submodule (geni-tools source code)
  cd docker-am
  cd geni-tools
  git submodule init
  git submodule update
  git checkout develop

5.2.3 4.2.3 Configure AM

Two files are used to configure the AM.

5.2.3.1 gcf_config

This file is for the generic GCF configuration. This contains the basic AM setup. The docker AM specific functionality is activated by setting delegate to testbed.DockerAggregateManager

Path of this file : docker-am/gcf_docker_plugin/gcf_config

- base_name: Typically, the name of your machine. This is the name used in the URN (urn:publicid:IDN+docker.example.com+authority+am)
- rootcadir: A directory where your trusted root certificates are stored. These are the certificates of the MA/SA servers whose users the AM will trust. (wall2.pem for example)
- host: This should be the DNS name of the server. It is used for binding the server socket. 0.0.0.0 is often a good choice if the hostname is not correctly configured on the server.
- port: You are free to choose a port. 443 is recommended (because it is infrequently blocked by client-side firewalls).
- delegate: To activate the docker AM code, this must be testbed.DockerAggregateManager
- keyfile and certfile = Private key and certificate of the AM server. This is used for SSL authentication. See the section below.

5.2.3.2 docker_am_config

This is the configuration that is specific for the docker AM.

Path of this file: **docker-am/gcf_docker_plugin/docker_am_config**

The [general] section currently contains one parameter.

- public_url: the URL to the AM, as advertised in the Getversion reply. This URL must contain the FQDN of the host. A raw IP address is discouraged. The following values for the hostname are forbidden here: 0.0.0.0 127.0.0.1 localhost

A [proxy] section is also allowed, but not mandatory (no automatic proxy is used if not specified). Check the example config for details.

Each other "section" in the config (a section start with [name_of_the_section]) in this file represents a DockerMaster (a dockermaster host one or more containers). If you want to configure multiple DockerMaster just duplicate the first section and change the name. Then, configure parameters:

- max_containers : The maximum number of container hosted by your DockerMaster
- ipv6_prefix: If you have an IPv6 address on your host, set the prefix in /64 or /80 (for example: 2607:f0d0:1002:51::) and each container will be assigned an IPv6 in this range
- dockermaster_pyro4_host, dockermaster_pyro4_password and dockermaster_pyro4_port: Parameters to connect to the dockermanager using pyro4 RPC (only when using a remote dockermanager, to use a local docker service, skip these options)
- node_ipv4_hostname: The IPv4 of your DockerMaster host (will be used to expose an SSH port on the containers)
- starting_ipv4_port: This is the first range used by docker for port forwarding. For example, if you set 12000, the first container should be reachable on port 12000, the second on port 12001, ... The AM uses the first port available from 12000 to 12000+max_containers

5.2.4 Configure a DockerMaster

You can run the docker service on either the same node as the AM, or use the remote DockerMaster feature (which uses pyro4 for RPC) to run the service on another node.

On the node where docker needs to run, do the following:

- First, install the docker engine:
  https://docs.docker.com/engine/installation/
- Then, make sure the daemon is running (with systemd):
  systemctl start docker.service
- If you want the docker daemon restart automatically after reboot:
  systemctl enable docker.service

See the section "Configuring a remote DockerManager (Optional)" for more details on a remote DockerManager

5.2.4.1 Configure IPv6 for Docker

If you want to use IPV6 on your container, you have to configure Docker bridge to use a specified prefix.

Create a new file (this path is available on Debian based distribution)

/etc/systemd/system/docker/service.d/docker.conf :
[Service]
ExecStart=
ExecStart=/usr/bin/docker daemon --ipv6 --fixed-cidr-v6="2607:f0d0:1002:51::/64" -H fd://

By replacing the IPv6 example by your own with the proper prefix length (64 or 80)

Then restart your docker daemon: systemctl restart docker.service

5.2.5 Generate certificate and key

You need to get a server key and certificate for the AM. You can either get a real one (using your regular way to get SSL Certificate, or using "Let's Encrypt"), or you can create a self-signed AM server certificate. In the latter case, you will need to add the self signed certificate to the trust store of all clients (which is not a big deal, as you need to add other server info anyway).

It is advised not to use the bootstrap-geni-am/geni-tools/src/gen-certs.py script provided by gcf, it does not generate a good AM server certificate. A valid server certificate should have:

- A Subject Name containing a CN equal to the server hostname
- One or more Subject Alternative Names of type DNS matching the server hostname(s)
- The server hostname mentioned in the 2 points above, should be a DNS name, never a raw IP address
- There is no real need for the certificate to contain a Subject Alternative Name of type URI that contains the URN of the AM. But it is off course no problem if it is included.

Also note that your AM server certificate has nothing to do with the root certificate of your cleaninghouse (= MA/SA). The clearinghouse root certificate is used for trusting credentials and for SSL client authentication, it has nothing to do with SSL server authentication!

To generate a self-signed server certificate, you can use the provided script:

cd generate-AM-server-cert/
./generate-certs.sh

Make sure the location of the server key and certificate matches the keyfile and certfile options specified in

bootstrap-geni-am/gcf_docker_plugin/gcf_config

You can find some more details on how a certificate can be generated at

https://stackoverflow.com/a/27931596/404495

5.3 ALLOW USERS TO USE THE AM (new since D2.9)

To allow users to use your AM, you need to add the root certificates that are used to sign these user's certificates to the AM's rootcadir. That is the directory that contains PEM files of all the trusted root certificates.

You can find and change this directory in gcf_config, by default, it is:

```
# The directory that stores the trusted roots of your CH/AM
# and those you have federated with
# This can be a relative or absolute path.
rootcadir=~/geni-tools/trusted_roots/
```

5.3.1 Trust your federation root authority

Each user authority will provide a root certificate in PEM form, that you can add to rootcadir.

For example if you add the root CA certiciate from Fed4FIRE to your testbed rootcadir, all users of the Fed4FIRE portal will be able to create experiments on your testbed. You can find the Fed4FIRE root CA certificate here: https://portal.fed4fire.eu/root_certificate

Example commands:

```
cd ~/geni-tools/trusted_roots/
curl https://portal.fed4fire.eu/root_certificate > portal_fed4fire_root_certificate.pem
# now restart your AM
```

5.3.2 Trust your C-BAS installation

If you use C-BAS as Member Authority (MA) and Slice Authority (SA), you have to trust credentials from this. To do, just copy certificates used by C-BAS to your rootcadir (configured in gcf_config). The certificates are usually stored in C-BAS/deploy/trusted/certs.

Example:

```
cp -v C-BAS/deploy/trusted/certs/*.pem ~/geni-tools/trusted_roots/
```

The restart your AM. If you check the output of the server you should have this at the beginning:

```
INFO:cred-verifier:Adding trusted cert file sa-cert.pem
INFO:cred-verifier:Adding trusted cert file ma-cert.pem
INFO:cred-verifier:Adding trusted cert file ch-cert.pem
INFO:cred-verifier:Adding trusted cert file ca-cert.pem
INFO:cred-verifier:Combined dir of 4 trusted certs /root/C-BAS/deploy/trusted/certs/ into file
/root/C-BAS/deploy/trusted/certs/CATedCACerts.p
```

5.4 STARTING THE AM

sh docker-am/run_am.sh

Or with the systemd service:

cp am_docker.service /etc/systemd/system/

Edit the WorkingDirectory according to your installation, reload the systemd daemon:

systemctl daemon-reload

then start the AM:

systemctl start am_docker.service

Check the status:

systemctl status am_docker.service

5.5 CONFIGURING A REMOTE DOCKERMANAGER (Optional)

You can set up several DockerManager hosted on different physical machine in order to increase scalability (for example).

5.5.1 Configure the remote

First of all you need to install dependancies on the remote host:

apt-get install python2.7 python-pip git
pip install pyro4

And install docker-engine:

https://docs.docker.com/engine/installation/

Now download the source code repository:

git clone https://github.com/open-multinet/docker-am.git

And try:

python2 docker-am/gcf_docker_plugin/daemon_dockermanager.py --host 127.0.0.1

You should get a warning about not using any password and a URI the server listening on.

You can use it in this way but it's more convenient to configure a systemd service. To do this, just copy the service file:

cp bootstrap-geni-am/dockermanager.service.sample /etc/systemd/system/dockermanager.service

Then edit the WorkingDirectory and ExecStart line in this file to match to your configuration. The "--host" parameter should be an IP reachable from the AM, so a public IP or, if your AM is on the same network a private IP.

Finally, do

systemctl daemon-reload && systemctl start dockermanager.service

and check with

systemctl status dockermanager.service

5.5.2 Configure the AM

On the AM, edit docker-am/gcf_docker_plugin/docker_am_config and add or edit a section to match the three parameters (dockermaster_pyro4_host, dockermaster_pyro4_password, dockermaster_pyro4_port) with the parameters set on the remote

Then delete am-state-v1.dat (to force configuration reload) and restart your AM.

5.6 HOW TO ADAPT THIS AM TO YOUR INFRASTRUCTURE?

If you want to test the AM with your hardware (not with Docker or in addition to Docker) you have to develop your own Python Class which manages your hardware.

You can follow the docker model as example. It is based on three classes: DockerMaster (dockermaster.py), DockerContainer (dockercontainer.py), and DockerManager (gcf_to_docker.py).

- DockerMaster is more or less just a pool of DockerContainers, because a DockerMaster should be a unique physical machine
- DockerContainer represents a single docker container, with some information like to ssh port, the IPv6, ...
- DockerManager is a generic class to manage docker from Python

So, if you want to represent a physical machine which can be reserved by a user the Python class should be a merge between DockerMaster and DockerContainer, you should inherit your class from ExtendedResource. This class is formed of all used methods by the AM, so you have to implement at least those methods (also have a look to geni-tools/src/gcf/geni/am/resource.py)

To kickstart coding this, the class "ResourceExample" is provided. It's dummy external resource manager, which can act as a starting template. You must write configuration processing code in testbed.py > _init_ to enable the resource. The dummy resource does nothing, to get started, edit the file resourceexample.py and find lines with ssh="exit 0;" and follow the instruction on the lines above. Note that the kickstart code assumes that your AM has SSH access to the external resource.

Once your resources are ready, you have to init them in testbed.py in the _init_ method by adding them to the aggregate configuration parsing. Be sure to delete am-state-v1.dat when testing, to force configuration reload.

Note: You should probably implement a generic wrapper for your infrastructure like DockerManager, it's easier to maintain, especially if you have different kinds of resources.

5.7 DEVELOPMENT NOTES

If you want to make some contribution to this software, here there is a quick explanation of each file:

- testbed.py: The aggregate manager main class, handle calls from the API
- dockermaster.py: Docker Master is a pool of docker container, it is called to get some DockerContainer instances
- dockercontainer.py: Represents a Container with methods to manage it
- gcf_to_docker.py: The DockerManager class, used as generic wrapper for Docker in Python, mostly used by DockerContainer
- resourceexample.py: A dummy resource to kickstart you to develop your own resource
- extendedresource.py: A generic resource class which adds some usefull methods to the base Resource class (which is in resource.py, in the geni-tools repo)
- daemon_dockermanager.py: The daemon used to create a remote DockerMaster using Pyro4 framework.

5.8 ADDITIONAL INFORMATIONS

- Objects are serialized in docker-am/am-state-v1.dat, so you can restart the AM without consequence
- Slivers expiration is checked every 5 minutes, and on each API call
- Warning: If you restart the host, docker containers are lost, to keep consistent state delete am-state-v1.dat before restarting the AM.
  - It will mostly work without deleting the file, but you could have some unexpected behaviors

5.9 TROUBLESHOOTING

- If you get the error "Objects specify multiple slices", you probably made a typo in component_manager_id (during allocate call)
- If your configuration is not taken in account, delete docker-am/am-state-v1.dat and remove all running containers docker rm -f $(docker ps -a -q)
- If you get an SSL error (like host not authenticated) check if you correctly add your AM/SA certs in trusted root

6 HOW TO JOIN THE TESTBED FEDERATION? (new since D2.9)

6.1 TYPES OF FEDERATION

There are 3 types of testbed relationships to Fed4FIRE:

- Associated testbeds: The testbeds are associated with Fed4FIRE but not federated. These testbeds are not supporting accounts of Fed4FIRE, nor Fed4FIRE APIs. These testbeds are just listed on the website of Fed4FIRE at https://www.fed4fire.eu/associated-testbeds/.
- Light federation: access to the testbed's resources is realized by exposing a Web-based API. This option does not necessary allow full control over the individual testbed resources, but ensures unified access to experimenters.
- Advanced: the testbed is fully integrated in the federation so that experimenters can interact with their experiment during all stages of the experiment's life cycle (resource selection, instantiation, control, monitoring, etc.). This option requires the implementation of the Federation AM API (Aggregate Manager Application Programming Interface) on top of your testbed.

6.2 LIGHT FEDERATION

The basic requirement here is that the users can use their Fed4FIRE account to access the testbed. This means that they typically cannot use Fed4FIRE tools or APIs to do this, but need to learn the specific testbed way of access.

Technically, Fed4FIRE supports the OpenID Connect layer on top of the OAuth protocol (https://openid.net/connect/) and this gives an idea of the metadata to be supported for the integration (https://portal.fed4fire.eu/.well-known/openid-configuration):


```
{
    "authorization_endpoint": "https://portal.fed4fire.eu/oauth/authorize",
    "id_token_signing_alg_values_supported": [
        "none",
        "RS512"
    ],
    "introspection_endpoint": "https://portal.fed4fire.eu/oauth/introspect",
    "issuer": "https://account.ilabt.imec.be",
    "jwks_uri": "https://portal.fed4fire.eu/.well-known/jwks.json",
    "response_types_supported": [
        "code",
        "id_token",
        "token id_token"
    ],
    "scopes_supported": [
        "openid",
        "userinfo",
        "privatekey",
        "slice_authority",
        "member_authority"
    ],
    "token_endpoint": "https://portal.fed4fire.eu/oauth/token",
    "token_endpoint_auth_methods_supported": [
        "client_secret_basic",
        "client_secret_post"
    ],
    "userinfo_endpoint": "https://portal.fed4fire.eu/api/userinfo"
}
```


6.3 ADVANCED FEDERATION

This is typically used for testbed resources that are accessible through ssh (including e.g. a testbed gateway that is accessible through ssh). The testbed is fully integrated in the federation so that experimenters can interact with their experiment during all stages of the experiment's life cycle (resource selection, instantiation, control, monitoring, etc.). This option requires the implementation of the Federation AM API (Aggregate Manager Application Programming Interface) on top of your testbed. You can find some possible ways to support the AM API at https://doc.fed4fire.eu/testbed_owner/amsoft.html without implementing this from scratch:

6.3.1 Lightweight AM interface

6.3.1.1 GCF - Geni Control Framework¶

Site: https://github.com/GENI-NSF/geni-tools

This python AM can be used to wrap existing testbed software. The GCF component "wraps" then the existing testbed API and makes a bridge between the testbed and the Fed4FIRE tools. Some testbeds (e.g. Grid5000) in Fed4FIRE used this approach.

Based on GCF, the docker AM example was created: https://github.com/open-multinet/docker-am. This AM provides resources using docker containers. Note that this is meant as a demo: the aim is not to offer the containers as actual testbed resources. This is a useful as a starting point, for testbeds that need to:

- Host resource management software in a container. Users will thus connect to this container to use the testbed resources.
- Provide access to resources through a "jump node" or "proxy" (which runs in a container).

Both are actively maintained and tested with the Fed4FIRE tools and accounts.

6.3.1.2 O2CMF

Site: https://gitlab.com/futebol/O2CMF

O2CMF is a framework based on GCF for federated experimentation based on the OpenStack Cloud Platform. It was developed in the Futebol-project. It provides an AM and a SSH Proxy.

6.3.2 Testbed management software with AM interface

This can be used if you are not bound to existing testbed software or if you still need to build your testbed.

6.3.2.1 Emulab

Site: https://www.emulab.net

Emulab is powerfull testbed management software, that includes support for hardware links. It is written mostly in perl, and is quite complex to install.

As several testbeds in Fed4FIRE use this software, it is fully supported and maintained.

6.3.2.2 GRAM - Openstack AM

GENI Aggregate Manager (AM) API front-end to a set of Openstack managed cloud resources

Site: https://github.com/GENI-NSF/gram/blob/master/RELEASES.md

This is not maintained anymore but might be a starting point.

6.4 FEDERATION APIs

For reference (you should try to re-use one of the above solutions to federate your testbed rather than implement the APIs from scratch), we add here an overview of the APIs and concepts used in the federation.

- Slice and slivers: a slice contains slivers. Think of a slice as your experiment which contains resources (slivers). Those slivers can be in different testbeds.
- Member Authority API (MA API): this API is used to interact with the authority about users and projects information and authentication. This API is based on XMLRPC over HTTPS. See also https://geni-nsf.github.io/CommonFederationAPI/CommonFederationAPIv2.html
- Slice Authority API (SA API): this API is used to interact with the authority about slice (experiment) information. This API is based on XMLRPC over HTTPS. See also https://geni-nsf.github.io/CommonFederationAPI/CommonFederationAPIv2.html
- Aggregate Manager API (AM API, https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html): this API is used to interact with the testbed. This API is based on XMLRPC over HTTPS. See also https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html
- Resource Specifications (RSpecs, https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html) describe the resources to be reserved/provisioned. They are defined in XML and come in 3 flavours:
  - Advertisement RSpec: can be retrieved from the testbed through the AM API and describes all available resources on a testbed
  - Reservation RSpec: This described the resources a user wants to reserve
  - Manifest RSpec: this resembles the Reservation RSpec, and is the RSpec that the testbed returns when a reservation is made. It contains e.g. information to access the reserved nodes.
- Some naming and identification concepts are defined as well, see some examples in the table below (URN = uniform resource name) (further information on this can be found at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/general.html):

Resource	ZZZZ Identifier
User cviecco at the planetlab namespace	urn:publicid:IDN+planet-lab.org+user+cviecco
Planetlab node: pl2.ucs.indiana.edu	urn:publicid:IDN+planet-lab.org+node+pl2.ucs.indiana.edu
Interface eth0 in planetlab node pl1.ucs.indiana.edu	urn:publicid:IDN+planet-lab.org+interface+pl1.ucs.indiana.edu:eth0
Slice mytestslice in the Utah Emulab slice authority	urn:publicid:IDN+emulab.net+slice+mytestslice
The Utah Emulab slice authority	urn:publicid:IDN+emulab.net+authority+sa
Sliver 123 in the Utah Emulab aggregate manager	urn:publicid:IDN+emulab.net+sliver+123

Figure 13: Naming and identification concepts

All these things can be easily verified and learned through the jFed user tool.

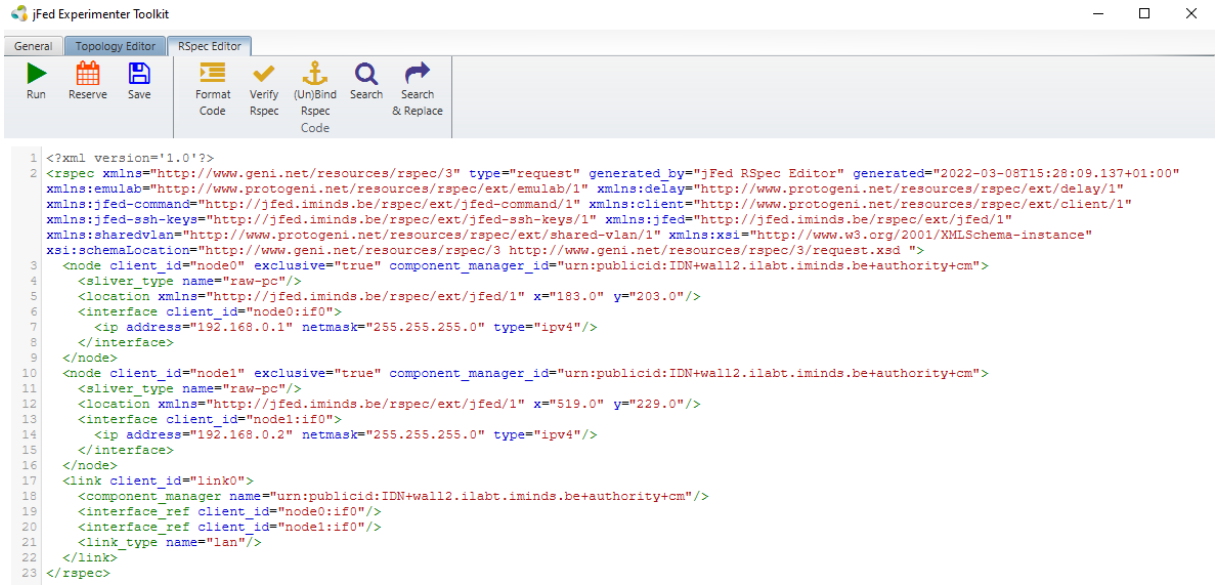


Figure 14: Reservation RSpec in jFed

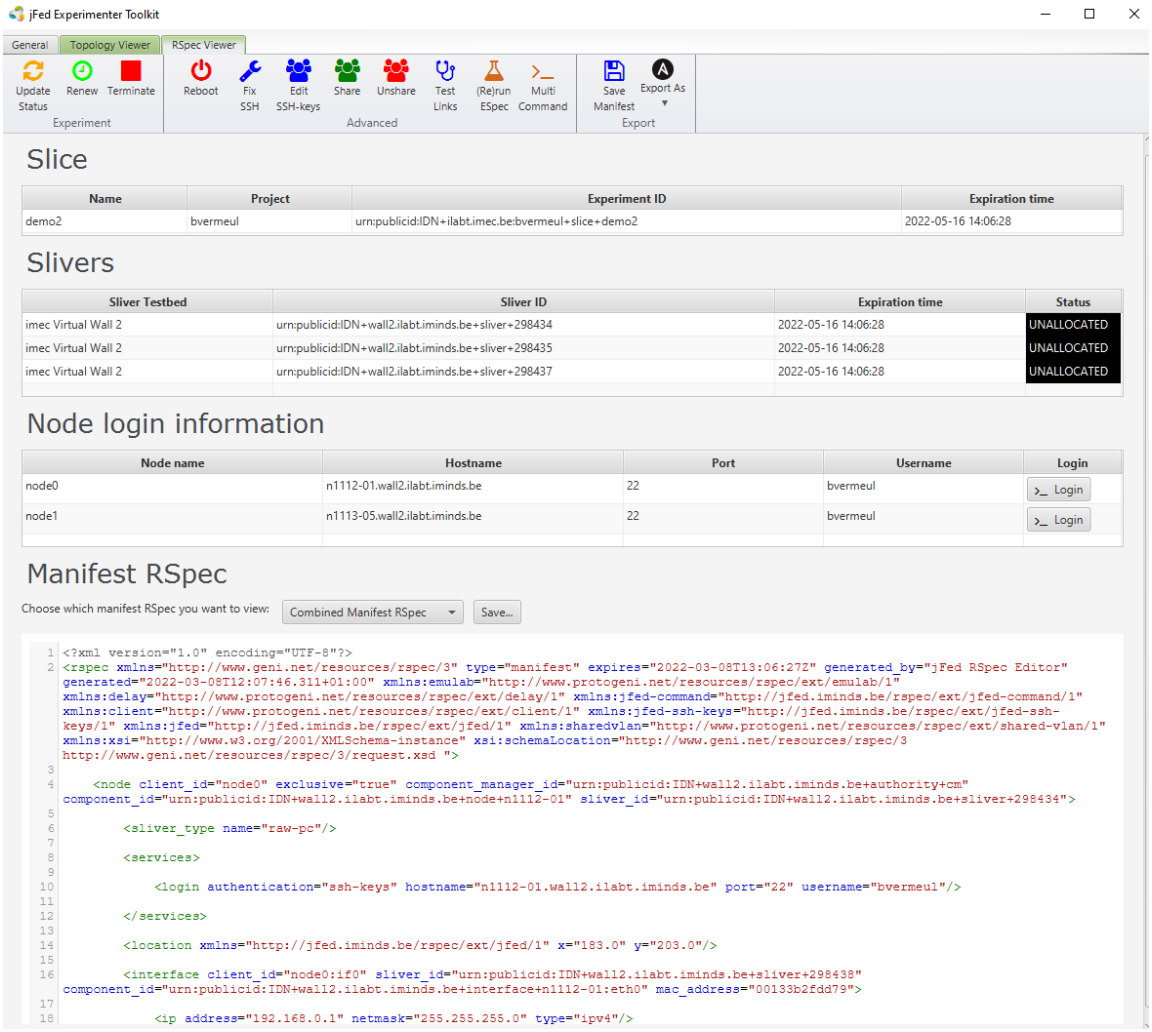


Figure 15: Manifest RSpec in jFed

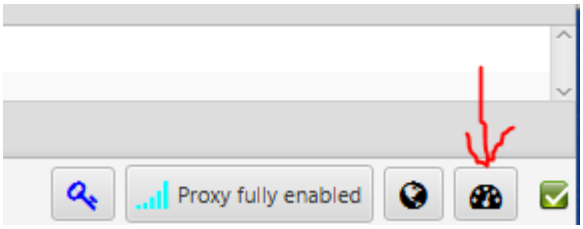


Figure 16: Use the call information button in the right bottom of jFed to access all API calls

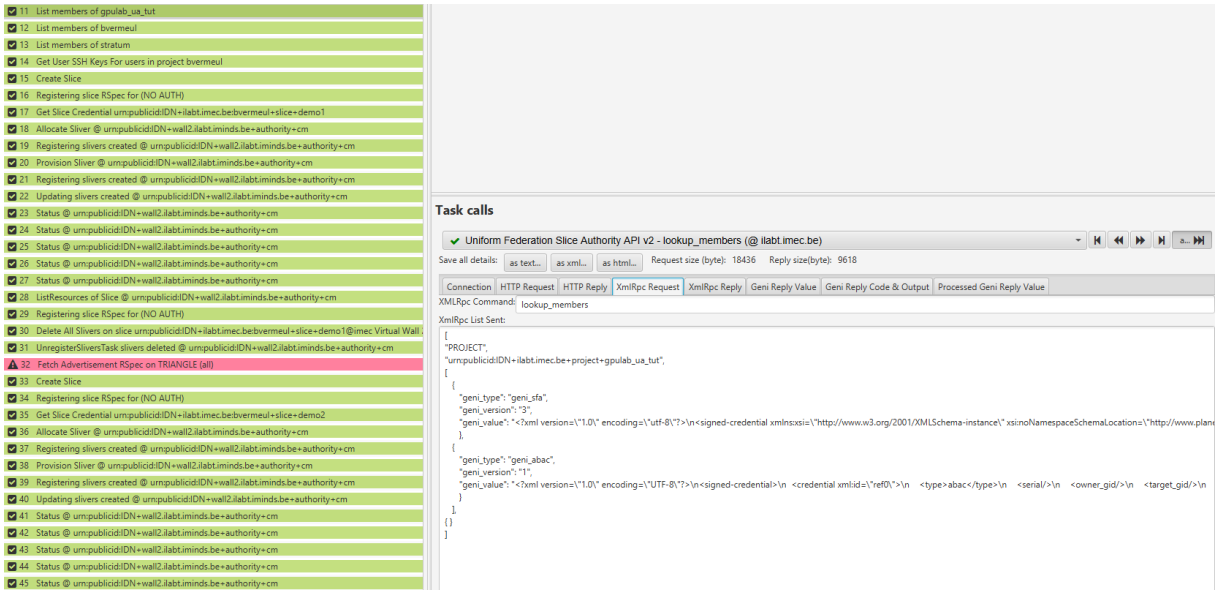


Figure 17: Example of API call in jFed (left shows all calls done, right shows specifically the lookup_members call). The calls can be verified at http and xmlrpc level

6.5 FEDERATING YOUR TESTBED WITH FED4FIRE+ THROUGH THE AM API

This section explains how to federate your testbed with Fed4FIRE+ through the AM API. For this we will use jFed.

First of all, note that this section is about adding an AM, an Aggregate Manager (sometimes also known as CM, "Component Manager"). If you want to setup and use your own user and slice authority (SA and/or MA), there are some extra complications not mentioned here. (Feel free to ask us for additional info if you are in this case.) Note that if you federate your AM with the Fed4FIRE+ SA and MA, there is no need to configure jFed to use the SA/MA of your testbed (if you have one), so there is also no need to configure it in jFed.

In short, this section assumes that you have a Fed4FIRE+ account, and you will login to jFed with this account, in order to use your AM.

6.5.1 Requirements

You need a few things before you get started:

- A server to run the AM software on.
- A publicly reachable IP for that server. This needs to be either an IPv4 or an IPv6 address. We recommend both.
- A DNS name for that server, that resolves to the publicly reachable IP addresses of the server. (Recommendation: It's nice if the DNS name refers to your testbed and is specific for your AM. Example: am.mytestbed.example.com)
    - Note:
      A DNS name is not strictly required, but very highly recommended
- Choose a URN for your AM. This is of the form: urn:publicid:IDN+DNSNAME+authority+am where you replace DNSNAME by the DNS name of your AM. (Example: urn:publicid:IDN+mytestbed.example.com+authority+am). If you don't have a DNS name for your server use a globally unique name for your testbed instead. It's not recommended to use an IP address instead, though that can work. Never use localhost or 127.0.0.1, and never use the default value of the AM software you use. (For more info, see Choosing your component manager urn)
- Choose a port at which you server will run. There is no standard port in the specification, so a lot of different ports are used in practice (12369, 8010, …). We recommend using port 443, if that is not in use by anything else. The advantage of using the default https port is that it is reachable through most firewalls, and the protocol is in fact using https.
- You need a X.509 Server Certificate, because the AM server uses https. This can be a self-signed certificate (jFed stores a list of these to make it work safely). However, in that case, make sure you configure the fields in your self-signed certificate correctly. See the next section for more details.
- You probably have testbed resources that you want to make reachable to experimenters using SSH. There are 2 options:
    - You need public IP addresses that you can assign to these nodes when needed (IPv4 or IPv6)
    - You need to have one machine with a publicly reachable IP address (IPv4 recommended) act as a gateway.

In both cases, make sure you have these address(es) available. You can combine the gateway machine with the machine on which the AM runs, but for security reasons, this is not recommended.

An example of these choices for the imec Virtual Wall2 testbed:

- IP: 193.191.148.179
- DNS name: wall2.ilabt.iminds.be
- URN: urn:publicid:IDN+wall2.ilabt.iminds.be+authority+cm So the URN including the DNS name as TLD/TLA part of the URN. "authority" as "type" of the URN, and "cm" as "name" part of the URN. (We actually recommend "am" as name instead of "cm", but both are fine.)

Server certificate:

Not self signed in our case, but signed by a trusted root.

- "Subject Name" contains CN=www.wall2.ilabt.iminds.be, as well as details about our address and organisation
- "Subject Alternative Name" contains DNS:www.wall2.ilabt.iminds.be, DNS:boss.wall2.ilabt.iminds.be, DNS:authority.ilabt.iminds.be

Our server runs at port 12369, because emulab already uses port 443 for another webserver, with different SSL settings.

The full URL at which our AM is reachable is:

https://www.wall2.ilabt.iminds.be:12369/protogeni/xmlrpc/am/3.0

Experimenters can reserve servers at our AM. All of these are then reachable with SSH using public IPv6 addresses. We also have a pool of IPv4 addresses which users can assign to servers if needed. In addition, we have a gateway (bastion.test.iminds.be, which has both an IPv4 and IPv6 address), which can be used by experimenters that do not have IPv6.

6.5.2 Server X.509 Certificate

An AM uses a message format specified in Geni AM specification, which is sent over XML-RPC, which is sent over HTTPS with client authentication. So an AM runs at a HTTPS server (= HTTP over SSL) that is configured to require client authentication.

This means that the firsts step is to correctly setup the SSL server. That requires a server X509 certificate and matching private key. This certificate is used to identify the server to the users. Because only the server has the private key matching the certificate, users can setup a secure connection.

For SSL on the public internet, it is required that your server's X509 certificate is signed by a trusted root. All browsers have a list of these trusted roots. This is in short how the internet is made secure.

jFed also uses the internet's trusted roots, so if your server certificate is signed by these, that should be enough. However, for convenience, jFed also allows you to use a self-signed X509 certificate. You can then add this self-signed certificate to the list of certificates that jFed trusts, and everything will work without requiring the hassle of aquiring a "real" X509 certificate signed by a trusted root.

However, in recent years, thanks to "Let's encrypt" and the excellent tool certbot it is now usually easier to get a real valid X509 certificate than it is to generate a decent self-signed certificate. Because of this, we recommend setting up certbot on your server, and letting it generate the X509 certificate. Make sure that you setup certbot to automatically renew the certificate (using cron or a systemctl timer), and to restart the AM server when it does (this can be done using the certbot "post-renew hook").

As mentioned, you can also generate a self-signed certificate and register it to the central list of certificates that jFed trusts. When generating a self-signed certificate there are some best practices to take into account, regarding the self-signed X509 certificate that you generate for your AM:

- "Subject" field of the certificate must contain a "CN" that is the hostname of the server (NOT and IP, the DNS hostname!)
- The "X509v3 Subject Alternative Name" section, must contain a "DNS" entry, which is the hostname of your server (NOT and IP, the DNS hostname!). Note that this means that your AM needs a DNS name, not just an IP address!
- If the self-signed certificate is signed by another self-signed certificate, you need to include that other certificate in the "certificate chain" when configuring the AM's SSL.

Note that you can also add your AM to jFed if the first 2 guidelines are not respected. However, try to at least have a meaningful "Subject" "CN", and avoid a "Subject" "CN" like "localhost" at all cost. Also not that this will probably cause problems with non-jFed tools.

6.5.3 Allowing access to federated users

6.5.3.1 Federate with Fed4FIRE+ User and Slice Authority

Your AM needs to be configured so it is federated with the Fed4FIRE+ user and slice authority. How to do this depends on the AM software.

General instructions: You need to add the new Fed4FIRE+ root certificate and optionally also the old Fed4FIRE+ root certificate to the list of trusted authorities.

AM specific instructions:

- For GCF, you need to put the root certificate file in the gcf trusted_roots directory (this is configurable with rootcadir in gcf_config, check the default value in the gcf repo). Example command to do this: curl https://portal.fed4fire.eu/root_certificate > ~/.gcf/trusted_roots/fed4fire.ilabt_imec_be.pem (and then restart the AM)

The result of configuring this is:

- Your AM will allow (client-authenticated) SSL connections from Fed4FIRE+ users. If not configured correctly, your AM might terminate SSL handshake from Fed4FIRE+ users (so no HTTP data is even sent). If correctly configured, your AM will know for each connection that a Fed4FIRE+ user is connected. This is the “authentication” part.
- Your AM will accept user and slice “credentials” that it receives from Fed4FIRE+ users. It will know what Fed4FIRE+ allows these users to do (typically, a slice credential will tell the AM a user can reserve resources). This is the “authorization” part.

6.5.3.2 Firewall

You’ll need to make sure some ports are reachable:

- Your AM is listening on one or two ports, and these should be publicly reachable.
- If your testbed nodes are reachable directly, the necessary ssh port on each node should be publicly reachable. If you use a testbed specific ssh proxy/gateway, only that proxy should be publically reachable.
- Your AM should be ping-able from the Fed4FIRE+ monitoring domain, so it should allow ICMP for at least that domain.
- Your testbed OML monitoring should be able to create outgoing connections to the Fed4FIRE+ monitoring domain. (Allow outgoing TCP to flsmonitor.ilabt.imec.be:3003)

Note: The Fed4FIRE+ monitoring connections will come from:

- 2001:6a8:1d80:2031:225:90ff:fe1d:1c68
- 193.191.148.194

Only for ICMP you could choose to allow only these (but you can also allow public access). For AM and SSH access, the ports need to be publicly reachable anyway. Don’t rely too much on these not changing, you probably need to allow the whole subnet.

6.5.4 Configuring the AM GetVersion call

Your AM needs to be configured to return the correct info in the GetVersion reply. This is not a critical step, and it can be skipped. But it is a good practice to get this right. It also makes the jFed scanner work better.

The **GetVersion** call is a call that the scanner uses to retrieve basic server info. It is defined in the AM specification. The results contain a lot of info, most of which will be automatically filled in correctly by the server software. Experience shows that the following fields have to be checked for correctness:

```
"urn": "urn:publicid:IDN+example.com+authority+am",
"geni_api_versions": {
  "2": "https://example.com/am/2",
  "3": "https://example.com/am/3"
```

You need to make sure that:

- The URN is correct. That includes the hostname part (example.com in the example above), and the name part am in the example above. The name should be am or cm. The URN must always match the expected URN in the component_manager_urn attribute in the request RSpec.
- The geni_api_versions URLs must point to the correct server URL(s). This must be the URL at which the AM is publically available. Make sure that:
- The protocol is https
- The port is correct
- The path is correct
- The hostname is not localhost or 127.0.0.1.
- Preferably, the hostname is not an IP address but a DNS hostname.
- Multiple URLs are allowed if the server supports multiple AM versions. But this is not required. Do make sure however that at least the version contacted is listed. That is, if you connect to https://example.com/am/2 it is at least expected that "2": "https://example.com/am/2" is listed.

6.5.5 Using the jfed scanner tool

Note: This section assumes you are using linux or MAC. Instructions for windows are very similar.

The jFed scanner can be used to automatically detect basic AM server settings and store them for use by the other jFed tools. This way, you can add use an experiment AM to your local jFed and test it.

While the scanner can be a useful tool, it fails in some scenarios (as it is hard to test all strange AM configurations). It is also not frequently tested, so it can break in some releases. Finally, it is not aimed at end users, and is thus not a user-friendly tool. For these reasons, if you experience problems with it, don't hesitate to contact us.

You can find the current stable jFed Scanner GUI in this archive.

After downloading the archive and extract it to a directory. Then execute scanner-gui.jar.

Example commands:

```
$ wget -nv https://jfed.ilabt.imec.be/releases/develop/VERSION/jar/jfed_gui.tar.gz
2017-01-10 11:27:28 URL:https://jfed.ilabt.imec.be/releases/develop/VERSION/jar/jfed_gui.tar.gz
[25654644/25654644] -&gt; "jfed_gui.tar.gz" [1]
$ tar xfz jfed_gui.tar.gz
$ cd jfed_gui
$ java -jar scanner-gui.jar
```

NOTE: Replace VERSION in the command above with a jFed version. You can find a link to the stable version at https://jfed.ilabt.imec.be/downloads/ or you can pick a development version at https://jfed.ilabt.imec.be/releases/develop/?C=N;O=D

Once the scanner tool has started, log in using your Fed4FIRE+ account.

In the next screen, fill in the **Aggregate Manager** URL, and click the **Scan** button.

Requirements for the URL:

- It needs to be HTTPS
- Do not forget the correct port
- Do not forget to add the full path to the AM endpoint
- Preferably, use a DNS hostname, not an IP address

When the scan is completed, you end up on the **Scan Overview** tab. Here, in the ideal case, you'll see 4 green **OK** statuses. If not, something went wrong. In the **Call Logs** and **Debug Logs** tab, you can find debug info which can help when something goes wrong.

In the **Scan Output** tab, all information the scanner gathered is shown. It is sometimes needed to uncheck the checkboxes next to any incorrect or unneeded information. But usually, you do not need to do anything here.

In the **Security** tab, you will find the X509 certificate of the server. You need to verify if this is indeed the certificate of your server, and if that is so, you need to check the checkbox next to **I trust all of the certificates above**.

Then, click on the **Show Results** button.

You will see a JSON snippet containing the info that jFed will use. You can edit the following fields:

- **id**: This should be a very short unique ID for the testbed. It should only contain letters, no spaces or special characters.
- **longName**: This is the "human readable" name of your testbed. This name may contain spaces and special characters.
- **server/name**: This is the "human readable" of the testbed server. This name may contain spaces and special characters. (It can be the same as the testbed longName)
- **defaultComponentManagerUrn**: This is the URN of your AM. Make sure it is correct.

If you click on **Add to local jFed**, the JSON snippet will be stored in **~/.jFed/extra_testbeds/<id>.json** (where **<id>** is the ID you chose for the testbed). All jFed tools will load files from that dir to add additional testbeds.

6.5.6 Local files for adding testbeds

Using local files, you can add your AM to jFed.

When the jFed scanner has succesfully scanned a server, it shows a button "**Add to local jFed**", which will write such a file.

These files are stored in the jFed config dir, in the **extra_testbeds** dir. On Linux this is in **~/.jFed/extra_testbeds/.**

Below is an example of such a file with dummy values. You can also look at the global jFed configuration (see link in next section) for real examples.

Content of **~/.jFed/extra_testbeds/mytestbed.json**:

```
{
  "id": "mytestbed",
  "longName": "My Brand New Testbed",
  "defaultComponentManagerUrn": "urn:publicid:IDN+mytestbed.org+authority+am",
  "allowLinks": false,
  "servers": [
    {
      "name": "My Brand New Testbed",
      "certificateChain": "-----BEGIN CERTIFICATE-----\nMII... SERVER CERTIFICATE HERE ...\n-----END CERTIFICATE-----\n",
      "urnTld": "mytestbed.org",
      "baseUrl": "https://am.mytestbed.org/",
      "flags": [
        "featureExecuteAndInstallService",
      ],
      "services": [
```

```
            {
              "api": "Geni.AM",
              "apiVersion": "3",
              "url": "https://am.mytestbed.org/",
              "urn": "urn:publicid:IDN+mytestbed.org+authority+am",
              "@type": "Service"
            }
          ],
          "defaultComponentManagerUrn": "urn:publicid:IDN+mytestbed.org+authority+am",
          "@type": "Server"
        }
    ],
    "@type": "Testbed"
}
```


6.5.7 Adding to jFed for all users

The jFed team can also add the same info about your AM, to the “global” jFed config. That enables the use of your AM for all jFed users. The global jFed configuration is available at

https://flsmonitor-api.fed4fire.eu/testbed?embed=true

Contact us to add your AM.

You do not need to have a local wokring config, but that helps.

The minimal info you need to send us is:

- The URL at which your AM runs (the full URL, including hostname, port and path)
- The name you want to see for your testbed inside jFed and at our monitoring site
- An example request RSpec (which shows us the sliver_type(s) used and other info)
- Does your testbed support (or even require) specify disk images in the request RSpec?
- Does your testbed support (or even require) assign specific nodes (component_id attibute) in the request RSpec?
- Does your testbed support (or even require) specify hardware types in the request RSpec?
- Are links between nodes of your testbed supported?
- Does your testbed support “stitched” links? (If you’re unsure what that is, your testbed doesn’t support them.)

Other info which is usefull to send us is:

- Which “icon” in the jFed experiment GUI is the best match for your testbed nodes (ex: physical node, VM, wireless, …)?
- For security reasons, it is recommended to also send your server certificate (in PEM format). We can also retrieve the certificate ourselves, but theoretically, that could be intercepted.
- The coordinates of the testbed (latitude, longitude and country)
- Info on the organisation running the testbed (full name, latitude, longitude, country, address, link to logo, link to website)
- Additional info about your testbed: link to documentation, link to testbed description and/or other general testbed information.
- If your testbed has a web interface (where users can login and control the testbed) in addition to the AM, you can send us that link as well.
- The email address(es) of the “primary” contact(s) of the testbed. If we need to contact someone about the testbed, we use these emails.
- The email address(es) of the “technical” contact(s) of the testbed. For technical questions about the testbed, these emails are used. These emails also are used for automatic mails about the tests, and in the future, testbed specific bugreports will also be sent here. Let us know whether or not you want the technical contact(s) of the testbed to receive automated emails when the testbed goes down.
- The email address(es) of the GDPR contact(s) of the testbed. Anything related to the GDPR we will send here.

- The URN(s) of the users that may have admin access to some of the data we store about your testbed. This is currently not used, but we plan on using this later to allow you to restart tests, enable/disable tests or edit testbed info. This is typically the URN of the users you use to login to jFed.
- Which software does your testbed AM use? Typical options are: emulab, gcf, openstack, …

(A lot of email addresses are mentioned above, but typically, they are all just the same single email address.)

6.5.8 Adding your AM to one of the experiment GUI "icons"

The jFed Experimenter GUI has a graphical editor, where you can drag "icons" to the canvas and in this way easily configure an experiment. Each of these icons has their own list of relevant testbeds.

Note that you do not need these icons to test your AM using the jFed Experimenter GUI: after you added your testbed to jFed, you can directly edit the XML Rspec, and everything will work.

It is currently not possible to manually add a testbed to one of these jFed icons in your local jFed install. Adding testbeds to the "icons" can only be done by the jFed developers. When you contact us to add your testbed to jFed for all users, we will discuss which icon(s) it needs to be added too and add it for you.

Typically, to allow you to test without adding the testbed to the icon for all users, we add it to "dev-testbeds". Icons added here are not available to all users. To show these testbeds in jFed, go to "Preferences", then select "GUI Editor" and enter "dev-testbeds" in the "Extra Flags" field. Now restart jFed, and you will be able to select the testbed when using the appropriate icon.

6.5.9 jFed and Testbed specific documentation

The jFed experimenter GUI shows links to the testbed documentation. This makes it easy for experiments to find testbed documentation.

Two links can be set in the central jFed config, which will both be shown in jFed:

- "Testbed Info" button: a link to a (short) general description of the testbed
- "Testbed Documentation" button: a link to a site that has step by step instructions on how to use the testbed from jFed. A short and simple "getting started" tutorial is ideal. It's off course also nice if this site contains links to more info and more advanced tutorials.

These links are stored in the central jFed config, and testbed owners can ask us to add/update it. In future jFed versions, this link might be shown more prominently.

6.6 MORE INFORMATION

More information, including debugging and monitoring tools can be found at https://doc.fed4fire.eu/#testbed-owner-documentation.

- Documentation how to start: https://doc.fed4fire.eu/#testbed-owner-documentation, see below
- Tools for testing the Aggregate Manager: jFed Probe and jFed Automatic tester (https://doc.ilabt.imec.be/jfed-documentation-5.9/otherjfedtools.html)
- An example/template Aggregate Manager with docker containers. This is used as well as front-end for an existing testbed. See https://github.com/open-multinet/docker-am
- Documentation on the AM API: https://github.com/open-multinet/federation-am-api and https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html
- Documentation on the Slice Authority and Member Authority APIs: https://github.com/open-multinet/CommonFederation-SA-MA-API and https://geni-nsf.github.io/CommonFederationAPI/CommonFederationAPIv2.html
- Continuous monitoring and testing, including detailed information when clicking through: https://fedmon.fed4fire.eu (see also Deliverable 3.2 for more information on this).

6.6.1 Getting Started

When you want to join Fed4FIRE with your testbed, you can start with the following webinar (and accompanying slides): Fed4FIRE WP2: info for new testbeds.

6.6.2 Documentation

- Creating your AM: software options
- Federating your testbed with Fed4FIRE+ through the AM API
- Handling Testbed Specific Terms & Conditions (i.e. handling GDPR)
- Debugging your AM
- Rspec details for AM developers
- Stitching details for AM developers
- Implementing webbased tool using speaks-for

6.6.3 Useful links¶

Other useful links for AM developers:

- Almost all AM servers use the Geni AM API version 3.
- There is also a more generic AM API, the *Federation AM API*, however, it is not completely finished, and it is not used by any servers or supported by any clients. Though if any servers appear, we will certainly add support to jFed. The API documentation is built from the github repo where you can discuss or change the API. (this documentation might be easier to follow than the GENI AM API v3, and it's the same API for 99%).
- Documentation on the Slice Authority and Member Authority APIs.
- More documentation on RSpec including info on the Fed4Fire RSpec extension for SSH Proxy.
- Use the Federation Monitor to monitor your testbeds status once the integration has been done

For more info, contact us at contact@fed4fire.eu

7 CONCLUSIONS

This deliverable provides a complete overview from the point of view of the testbeds on the requirements, developments, and integrations. This implies different parts and types of work to be done. The work described here covers the whole period of the project and is therefore a final version of the series of deliverables D2.04, D2.09 D2.13.

The document provided an overview of the testbeds which have joined the federation since the start of the project. Most of them are external testbeds, but some of the testbeds are new testbeds at Fed4FIRE+ partners or further integrations of the testbeds at the Fed4FIRE+ partners.

All the testbeds at the Fed4FIRE+ partners provided feedback on how Fed4FIRE+ and belonging to the federation meant added value to them. But the testbeds also "listen" to their users and suggestions / comments made by individual experiments form the Open Calls which triggered specific actions / modifications / updates at the testbeds are also provided in detail.

A large amount of work was dedicated to set up a proper and fluent process to federate new testbeds into the Fed4FIRE+ federation. Information on this is provided and described in detail in this document and with a process, codes and links to supporting information on how to join the federation.

This document and process is now going to be used for the new testbeds joining as a result of the Open Call on new testbeds.