Federation tools: An Island Connectivity Experiment with Community-Lab

Gerard Marin, Leandro Navarro
Department of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, Spain
Email: {gmarin,leandro}@ac.upc.edu

Bart Braem, Chris Blondia
Department of Mathematics and Computer Science
University of Antwerp - iMinds
Antwerp, Belgium
Email: {bart.braem,chris.blondia}@uantwerpen.be

***Abstract*—This paper describes the standard or testbed-agnostic interfaces offered by Community-Lab an exemplifies its use in a reference experiment, the Island Connectivity Experiment. The experiment provides a web application that shows a matrix of network measurements across a slice of testbed nodes from different community networks in the testbed. The experiment uses testbed-agnostic and interoperable interfaces such as the Slice-based Facility Architecture (SFA) for resource management, resource description (RSpec), experiment management framework (OMF), instrumentation and measurement collection (OML), and presents a web interface. The experiment uses SFA to selects nodes for a slice, runs the measurements using OMF and collects measurements using OML. The experience in running the experiment validates the implementation of these testbed-agnostic interfaces, shows the complementarity and correct integration among the tools and interfaces as part of the Fed4FIRE federation, and confirms the value of a federation of testbeds and the usage of testbed-agnostic tools for experimentation.**

Keywords*—*Community-Lab.net, Experiment, testbed-agnostic, SFA, OMF, OML.

I. Introduction

Community networks (CN) are an emerging and successful model for the Future Internet across Europe and far beyond. In this model, citizens and organisations pool their resources and coordinate their efforts to build network infrastructures. The coverage of underserved areas and the fight against the digital divide are the most frequent driving factors. Technologies used vary in great manner, ranging from very low cost off-the-shelf wireless (WiFi) routers to expensive optical fibre (OF) equipment [1].

Models of participation, organisation and funding are very diverse but community networks are characterised for being open, free and neutral: open to know how they are built, free because the network access is driven by the non-discriminatory principle, and neutral both in terms of technology as long as network interoperability is respected to extend the network, and neutral for traffic from any kind of participant.

Representative examples[1] are Freifunk (FF) in Germany, the Athens Wireless Metropolitan Network (AWMN) in Attica, Greece, FunkFeuer (0xFF) in Austria, Ninux.org in Italy, Wireless België in Belgium, Sarantaporo in Greece, or guifi.net in Spain.

Although CNs have already been studied from several angles [1] [2], there is still insufficient understanding of the practises and methodologies which have given rise to such complex collaborative systems and these network infrastructure face several challenges that are being addressed by systems and networking researchers, hardware and software developers, and service providers.

The FIRE initiative (Future Internet Research and Experimentation) offers an open research environment which facilitates strategic research and development of new Future Internet concepts, giving researchers the tools they need to conduct large-scale experiments on new paradigms.

The CONFINE project[3] complements existing FIRE infrastructure by establishing the Community-Lab.net testbed built on the federation of existing community IP networks constituted by more than 30,000 nodes and 50,000 Km of links. The project develops a unified access to the testbed with tools that allow researchers to deploy, run, monitor and experiment with services, protocols and applications on real-world community IP networks referenced before. This integrated platform provide user-friendly access to these emerging networks supporting any stakeholder interested in developing and testing experimental technologies for open and interoperable network infrastructures, strengthening open community networks. The Fed4FIRE project[4] has established a common federation framework for experimental testbeds across the FIRE initiative with standard interoperable services and tools that support testbed-agnostic experiment life-cycle management, monitoring and trustworthiness.

The main contribution of this paper is the description of the environment, process and tools required for performing experiments in the Community-Lab testbed using testbed-agnostic interoperable mechanisms and tools. This is illustrated and validated by a complete reference experiment as a validation test. We also provide an overview of possible challenges and issues encountered when performing experiments with these testbed-agnostic tools.

The remainder of this paper is organized as follows.

[1] FF: http://freifunk.net/, AWMN: http://www.awmn.net/, 0xFF: http://www.funkfeuer.at/, Wireless België: http://www.wirelessantwerpen.be, Sarantaporo: http://www.sarantaporo.gr, guifi.net: http://guifi.net

978-1-4673-8103-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FiCloud.2015.107

An analysis of the federation scenario in Community-Lab is presented in section II. Section III describes the Island Connectivity Experiment (ICE) that measures the connectivity among testbed islands but also tests the behaviour and validity of the set of standard tools and interfaces offered. The results of the experiment are described in section V. Finally, after an overall discussion in section VI the paper concludes in section VII.

II. The Federation

Federation is a mechanism to enable the interoperability of facilities so they can work together, become a part of a larger infrastructure for experimentation, and offer generic APIs to support generic experimentation tools [5]. The three main aspects when preparing and performing an experiment are:

- Resource management: the selection, allocation, configuration and usage of a set of resources required for an experiment that can be drawn from a given resource aggregate, a testbed.
- Experiment management: the coordination of actions to accomplish the goals of the experiment using available resources. This includes planning and controlling the execution of the different steps in an experiment.
- Measurement collection: the instrumentation, measurement and collection of relevant measurement data according to the experiment description.

Each aspect is described as follows:

A. Resource Management

The Community-Lab testbed is federated with other testbeds according to the Slice-based Facility Architecture (SFA). SFA [6] is an architectural model to support testbed federation. SFA basically addresses the resource allocation procedure, defining standard interfaces and protocols that can be used to allocate resources in the different testbeds of a federation domain in an homogeneous way.

SFA is based on a set of high-level concepts that define the actors and the resources that interact on the testbed, as well as defining an architecture with its interfaces and main data-types to facilitate the federation of testbeds. SFA considers three principals: i) the management authority, responsible for a set of physical components, ensuring that the components behave properly and execute the resource allocation wishes of the component owner; ii) the slice authority, responsible of one of more slices; and iii) users, who are people playing one or more role in the facility.

The resources managed on a testbed are not only the physical substrate, but also the share of resources assigned to a researcher, which usually correspond to virtualized versions of the former substrate. SFA abstracts those resources in: i) Components, which represent the minimal aggregation of physical resources that can be managed; ii) Slivers, which are the portion of such resources let to the researchers; and iii) Slices, which are collections of slivers assigned to researchers to perform an experiment. Slices are the primary abstraction for accounting and accountability. SFA defines three stages in a slice life-cycle: (i) register: at this point the slice exists only in name; (ii) instantiate: the slice is instantiated in the required components, being granted of a set of resources; and (iii) activate: the slice becomes active and runs code on behalf of the researcher.

The software modules that manage those components are the aggregate manager for the components (or component manager if it manages a single component) and the slice manager for slices and slivers.

SFA also defines a set of main data-types:

- Global identifier or GID is an identifier assigned to components, slices, services and every principal participating in the system. Specifically a GID is a certificate that binds together a public key, a UUID and a period of time during while the GID is valid.
- Resource specification or RSpec is used to describe the resources on the system. The RSpec is an XML description of the testbed, since every testbed will have its own type of resources and requirements.
- Ticket is a promise signed by an aggregate manager, giving an entity the right to allocate the resources that are being granted.
- Credential, on the other hand, is a grant of a set of rights and privileges associated with a particular principal.

The Community-Lab testbed supports SFA. A software component called C-Lab SFA Wrapper adapts the testbed-specific procedures to the SFA standard and exposes the required interfaces[7]. Using an SFA client tool, a user can interact with the C-Lab SFA Wrapper to allocate resources in the testbed, that is, to create slices and slivers. To describe the resources that will be allocated, an RSpec document is used. An RSpec is an XML document used in SFA to describe the resources in the resources allocation process.

B. Experiment Management and Measurament

OMF is a framework to support experiment definition and management in testbeds [8]. The framework provides a language to define experiments, as well as two different entities that allow the installation and deployment of the defined experiment in the resources of the testbed. The idea is very simple: once the experiment is defined, the user uses an OMF server (Experiment Controller) to install the experiment and deploy it in the slivers of the testbed. The OMF server will control the corresponding slivers. Each sliver of a testbed supporting OMF will have an OMF client process (Resource Controller) running that will receive the messages from the OMF server and perform the required actions to install and execute the experiment.

OML is an instrumentation framework to support measurements and monitoring in testbeds [9]. The framework allows to define measurement points to collect specific data. Although the OML framework can be used stand-alone, it is usually used together with the OMF framework.

To understand the definition process it is important to distinguish between the OMF experiment and the OMF application. The OMF application is basically a Ruby Gem

application that wraps some system command (for example the "ping" command) to generate some OML Measurement Points in order to return the desired results. An OMF experiment is an experiment that uses an OMF application. The definition of the OMF experiment is given to the OMF EC host (server) that communicates with a set of specified OMF RC hosts. The OMF RC processes will receive messages from the EC (server) and invoke the required OMF applications, sending the results of application back to the server. This results will be OML streams coming from the OML Measurement Points of the application. The OML server specified in the experiment definition will receive the OML streams with the results of the experiment and store them in a database backend.

The Community-Lab testbed supports OMF also with OML included. In order to use OMF in the slivers of the testbed, the user needs to choose a specific template that includes all the OMF packages and dependencies. Actually, the template includes a modified version of OMF that supports IPv6 addresses, in order to use the IPv6 Management Network Overlay of Community-Lab for OMF communication purposes. The slivers created with this template will run the OMF Resource Controller process. The host acting as a OMF server, that is, Experiment Controller, can be another sliver of the testbed or any other machine with connectivity to the Resource Controller slivers. To use a sliver as Experiment Controller, the user needs to choose a specific OMF server template available in the testbed.

III. The Experiment

The Island Connectivity Experiment (ICE) is an experiment deployed in the Community-Lab [3] slivers that implements a long-running service to monitor the connectivity among the islands that form the testbed.

Basically, the slice of the experiment contains several slivers hosted by nodes in the different testbed islands embedded in diverse community networks [1]. These slivers periodically ping each other using their public IPv4 address, so that the connectivity among them is checked. Such pings give information about the reachability of the slivers in different islands from a sliver in a specific island. This information can be taken as a first approach to monitor the connectivity among islands.

Apart from the usefulness of the island monitoring service, this experiment also aims at using and testing the different federation technologies that have been developed in the Community-Lab testbed as a contribution to the Fed4FIRE project[4]. This experiment makes use of these technologies rather than the specific tools provided by the Community-Lab testbed, validating their feasibility and enabling evaluation of their performance and robustness.

Because of the federation complexity, as described in section II, a number of consecutive configuration and deployment steps are required to setup, run and analyze an experiment which is controlled by federation tools. The idea of this work was to integrate all the necessary steps to deploy the ICE experiment into a script. Once invoked, the script will automatically perform all these steps making usage of the corresponding tool or protocol. In other words, the script will encapsulate the experiment from the beginning to the end. At a more general level, this script is an abstraction of the general steps required for experiments on federated infrastructures.

The experiment consists of three phases, each using a different federation tool, as described in the following sections:

1) Resource Allocation with SFA and RSpec.
2) Experiment definition, deployment and execution with OMF.
3) Measurement results collection with OML.

A. Resource allocation with SFA and RSpec

In the Experiment, the slice and its slivers are created using a SFA client tool. In particular, the RSpec used in the experiment is a C-Lab Request RSpec that includes specific elements and tags to support the customization of Community-Lab slices and slivers and their specific features.

B. Experiment Definition, Deployment and Execution with OMF

The experiment that implements the long-running service to monitor the island connectivity is defined using OMF. In OMF the experiment basically consists of a wrapper around the classic ping application that receives the list of sliver addresses to ping. The experiment definition is given to the sliver (or external host) that acts as OMF Experiment Controller. This sliver is responsible for installing and executing the experiment in the other slivers.

C. Measurement Collection with OML

In this experiment, the OML measurement points are defined inside the experiment definition. After the experiment execution the collected data will be sent to an OML server for storage in an SQL database back-end.

IV. Implementation details

In what follows we will describe a number of tools and concepts which helped realize the ICE experiment. More complete documentation including detailed configuration files are available at http://wiki.confine-project.eu.

A. Topology

The topology used in the Island Connectivity Experiment consists of a slice registered in the Federation Authority of iMinds (Emulab). Although in this experiment the resources will only be allocated in one single testbed, the Community-Lab testbed, this step is necessary for using the SFA client tool to allocate the resources.

On the Community-Lab testbed side a slice for the experiment is created, as equivalent slice for the slice registered in the iMinds authority. The slice contains the following slivers as shown in figure 1: 1 sliver with the OMF EC template in a specific node from the UPCcloud island. The node is chosen to be stable and with enough available resources. 12 slivers with OMF RC templates: 2 slivers per island, in random nodes. We let the SFA Wrapper randomly choose the nodes, only imposing the condition of 2 slivers per island.

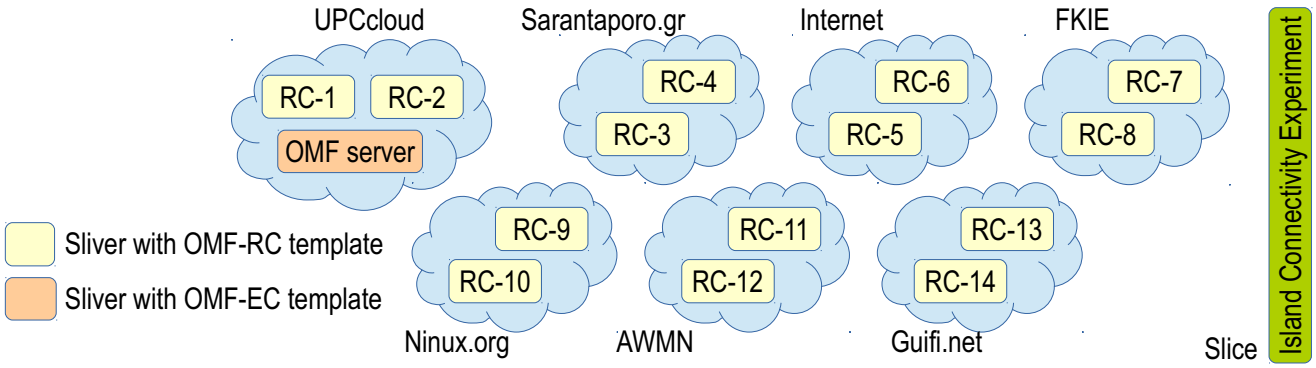

Fig. 1. Diagram of the ICE experiment

B. Resource details with Confine-ORM

Confine-ORM, which stands for Confine Object Resource Mapper or Object REST Mapper is a high level Python library to easily interact with the Confine REST API using object oriented concepts[2]. The library encapsulates all the interaction with the REST API of the Community-Lab testbed and allows the user to communicate with the testbed controller and the registry API to perform any operation. In the case of the ICE experiment, this library was used to get information about the slivers created in the Community-Lab testbed, such as network information.

C. Resource management with Omni

Omni is a command line tool for interacting and reserving resources at GENI Aggregate Managers (AMs) via the GENI AM API[3]. In other words, it is an SFA command line client. To use the Omni tool a user account issued by a GENI clearinghouse is required. The Omni client tool communicates with a user's clearinghouse (the federation authority) to create slices. Then it uses user and slice credentials to reserve resources from the available aggregate managers.

D. Resource allocation

In order to realize the fully federated experiment, the following steps are followed, starting by the SFA resource allocation with SFA. This process is performed using the Omni client tool to interact with the federation authority and the Community-Lab testbed through its C-Lab SFA Wrapper:

1) *Create slice in the federation authority*
   The first step in the resource allocation procedure is to create a new slice within the federation authority domain, in this case the Wall2 iMinds Authority. When this slice has been allocated, slivers for this slice can be created in any of the federated testbeds of the federation authority domain.
2) *Allocate the resources described by the RSpec in the C-Lab testbed*
   Then we create the required slivers for the new slice in the Community-Lab testbed. The slivers are described in the already describe RSpec format. The slivers are created calling the Allocate operation in the Aggregate Manager interface of the C-Lab SFA Wrapper. This operation will first create an equivalent slice in the testbed and then registers the new slivers described in the RSpec file.
3) *Set the state of the resources to "start" to begin the allocation process*
   The last step is to perform the operational action "start" in the allocated resources. To attain this result, in the background state transitions for the consecutive allocation states are performed (Allocate, Deploy, Start). Because these transitions trigger the actual allocation processes, a delay will be involved. As a consequence, the experiment scripts include code to efficiently wait for the desired intermediate allocation states, using the SFA Describe operation on a slice.
4) *Get CLab RSpec of the allocated resources to get information about the slivers*
   Once the resources are ready, the experiment uses the Omni tool to retrieve information about the slivers. It invokes the Describe operation on the created slice, but this time using the C-Lab RSpec type. Therefore, the Aggregate Manager of the C-Lab SFA Wrapper will reply with an extended RSpec containing some specific C-Lab elements and tags that will give complete information about the created slivers, e.g. including the internal IPv6 addresses.
5) Run the experiment itself to configure the slivers and start the execution of the OMF experiment
   In this final allocation step, we configure all slivers and start the necessary processes to perform the OMF experiment.

E. Experiment execution with OMF

Once all resources have been allocated, they will be used by OMF to perform the experiment itself. In order to do this, the gathered information about the slice and slivers (e.g. IP addresses) will be used in the following steps.

1) *Get the information of OMF Experiment Controller server*
   One sliver will act as the OMF EC, to manage the entire experiment. This information is extracted from the SFA information.
2) *Configure the OMF Resource Controllers*

[2]Confine-ORM documentation: http://confine-orm.readthedocs.org/
[3]Omni wiki: http://trac.gpolab.bbn.com/gcf/wiki/Omni

	AWMN-CF-7net	AWMN-Giannis1-NUC-VM	GB-MNJoanXXIII
AWMN-CF-7net		100.967ms 100.0%	104.66ms 0.0%
AWMN-Giannis1-NUC-VM	? 100.0%		? 100.0%
GB-MNJoanXXIII	110.695ms 0.0%	5.246ms 100.0%	

Fig. 2. Result of the ICE experiment: excerpt of the connectivity matrix

All slivers besides the OMF EC act as OMF resource controllers. To configure them, the individual configuration of each OMF RC is uploaded to the sliver. This includes information such as the AMQP URI, the OML server URI and all required OMF files (e.g. ping scripts) which are copied to the RC.
In this step the RC itself is started with the experiment configuration, including the IPv6 address of the OMF EC sliver.

3) *Prepare the OMF experiment file*
In this step the experiment files are customized to use the particular experiment parameters, e.g. setting the OMF EC IP address, the OMF RC sliver IP addresses and the number of pings.
4) *Configure the sliver that acts as OMF EC server*
In this step the configuration file for the OMF EC itself is prepared, including indicating involved slivers.
5) *Start the OMF experiment in the OMF EC*
At this point the slivers are correctly configured and provided with all the files needed to run the OMF experiment. The last step is to start the OMF experiment. To do that, the OMF EC is started with the prepared experiment configuration.

At the beginning of the experiment the OMF EC sliver will send messages to the OMF RC slivers with the description of the experiment and the tasks that they need to perform. The OMF RC slivers will start running the experiment, which basically consists of periodically sending pings to each other and exporting the result of these pings as OML streams. The OML streams are received by the OMF EC server that will store the data in its own OML server or forward it to the external OML server previously specified.

V. Experimental results

An execution of the experiment can result in a matrix such as the excerpt presented in figure 2. The labels indicate the sliver names, the table entries show the average ping times and ping loss percentage. At first sight the experiment seems to work fine. However, we observed some strange outcomes when the experiment has run for a while, indicated by the question marks.

The slivers send the results to the OML server through the Community-Lab IPv6 Management Network overlay. At some point however, there are nodes that lose the connectivity with the OML server. This implies that, although the nodes perform the experiment and ping all the other nodes, the results of these pings are not exported to the OML server. In the logs of the OMF RC of the nodes, it can be seen that the nodes are actually running the experiment correctly, but the results are not being sent to the OML server due to a loss of communication. The node tries to reestablish the communication and, according to the log, the connection seems to be reestablished eventually. However, the nodes do not send the results to the server again. We expect this to be an OML bug.

This strange behavior causes a lack of results availability in the OML server. Therefore, when looking at the results of the experiment, the matrix that shows the connectivity among some nodes is barely populated. Many cells show the message of “No info available”.

Note that this behavior happens after a while of running the experiment. At the beginning of the experiment, that is, up to the third or fourth iteration, most of the nodes seem to report the results correctly to the OML server. Further investigation is required to solve this problem.

Although this kind of bugs are disappointing, it illustrates the kind of challenges involved in federated experiments. All steps in the experimental process, including resource allocation, experiment execution and result gathering, are vulnerable to potential bugs and issues.

VI. Discussion

The tradeoff lies between genericity in testbed-agnostic tools and the specificity in testbed specific one. Genericity brings the promise of wider applicability and reuse but, as seen in the ICE experiment, it also comes at the cost of learning the details and using these generic tools that require the experiment to understand, decide and go over several minor but annoying details. While simple experiments focused on a single testbed may not benefit, larger experiments that can leverage on mature tools, that manage and hide complex details, can definitely see a big advantage. Testbed-agnostic tools can be particularly beneficial for experiments focused on repeatability and interchangeability, either on different testbeds, different parameter setups or performed by different researchers.

The learning curve for researchers is still high. Networking and systems researchers tend to have good scripting and programming skills and tend to develop their own do-it-yourself low-level and low-cost tools. The long experience with experimental research using testbeds such as PlanetLab, Orbit or our own Community-Lab and WiBed illustrates this. In fact SFA emerged from the long experience of one testbed and it was developed over the years to facilitate integration (federation) with other similar testbeds. However, the promise of learn once and use in all testbeds is very enticing.

The “unbundled management” principle [10] in PlanetLab shows there are benefits in diversity, evolution and competition: functionality that can be implemented by parallel, competing subsystems, versus mechanisms can only be implemented or offered once. This preference for services to compete with each

other on a level playing field shows that native and agnostic experimentation tools can coexist.

A key factor for adoption is in how these generic tools can facilitate experimentation and free experimenters in dealing with details that are not relevant for the experiment but that can affect it. One aspect has to do with the maturity of the tools: as the software implementation of tools and their "testbed drivers" becomes more robust and polished, researchers may not need to worry about low-level details and write simpler experiments. Conversely some features are not yet offered by agnostic tools because they are simply not yet implemented, or because they are too hard to express in the common testbed-agnostic model, exemplified by the adoption of semantically rich resource descriptions such as RDF or OWL that appears inevitable. Another is the conceptual aspect: it's easy for researchers to think in terms of a common conceptual architecture (SFA) and learn only about how to deal with the specific details of a given testbed. If fact, current community efforts, exemplified by the FIRE and GENI research initiatives, are developing sophisticated and mature enough experimentation tools that may become soon the mainstream way to perform systems and networking experiments.

For testbed developers and providers, in our experience in the implementation of SFA and OMF support in Community-Lab, the effort required to develop its native interface is comparable to the effort required for the development of the testbed-agnostic interfaces. In fact, in Community-Lab the testbed-agnostic interface is built on top of the native interface. This means we had to implement both. Given the majority of our experimenters preferred the native interface, easier for small experiments, the agnostic interface does not implement all the features that the native has included based on feedback from experimenters.

VII. Conclusion

This paper describes the standard specifications and interfaces offer by the Community-Lab testbed as part of the Fed4FIRE federation. It describes how the set of tools that have been developed and adapted to Community-Lab allow the experimentation life cycle using testbed resources. This is based on the Slice-Based Architecture (SFA), RSpec, Aggregate Manager (AM), cOntrol, measurement and Management Framework (OMF/OML).

The Island Connectivity Experiment (ICE) has been developed, deployed and executed in a slice of Community-Lab resources using these standard protocols and tools. The experiments performed validate the Community-Lab testbed as a standards compliant testbed, completely integrated in the Fed4FIRE federation of FIRE testbeds. The main lesson learned is the trade-off between the simplicity and flexibility of using native non-standard and testbed specific tools, and the complexity and formality of using standard and interoperable testbed-agnostic tools. We believe that while simple experiments may benefit from the first, more complex experiments, and particularly those that require multiple testbeds, are enabled by these testbed-agnostic formats, interfaces and tools.

Acknowledgment

This work is supported by European Community Framework Programme 7, the FIRE Fed4FIRE project, the "Community Networks Testbed for the Future Internet" (CONFINE), and the Spanish government contract TIN2013-47245-C2-1-R.

References

[1] J. Avonts, B. Braem, and C. Blondia, "A questionnaire based examination of community networks," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2013.

[2] L. Maccari, "An analysis of the ninux wireless community network," in *9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2013, Lyon, France, October 7-9, 2013*. IEEE, 2013, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/WiMOB.2013.6673332

[3] B. Braem, C. Blondia, C. Barz, H. Rogge, F. Freitag, L. Navarro, J. Bonicioli, S. Papathanasiou, P. Escrich, R. Baig Viñas, A. L. Kaplan, A. Neumann, I. Vilata i Balaguer, B. Tatum, and M. Matson, "A case for research with and on community networks," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 68–73, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500098.2500108

[4] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, M. Sioutis, A. Quereilhac, Y. Al-Hazmi, F. Lobillo *et al.*, "Architecture for the heterogeneous federation of future internet experimentation facilities," in *Future Network and Mobile Summit (FutureNetworkSummit), 2013*. IEEE, 2013, pp. 1–11.

[5] G. Marin and L. Navarro, "Federation of community networking testbeds," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, Oct 2014, pp. 200–204.

[6] "Slice-Based Federation Architecture," Version 2.0, GENI Initiative, Tech. Rep., Jul. 2010. [Online]. Available: http://groups.geni.net/geni/wiki/SliceFedArch

[7] G. Marin and L. Navarro, "Federation of community networking testbeds," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*. IEEE, 2014, pp. 200–204.

[8] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "Omf: A control and management framework for networking testbeds," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, pp. 54–59, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1713254.1713267

[9] M. Singh, M. Ott, I. Seskar, and P. Kamat, "Orbit measurements framework and library (oml): motivations, implementation and features," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*, Feb 2005, pp. 146–152.

[10] L. Paterson and T. Roscoe, "The Design Principles of PlanetLab," *Operating Systems Review*, vol. 40, no. 1, pp. 11–16, January 2006.