Heterogeneous Resource Provisioning for Cross-Layer Future Internet Testbeds

Konrad Campowsky[1], Bogdan Harjoc[2], Thomas Magedanz[1,2], and Sebastian Wahle[1]

[1] Fraunhofer FOKUS, Next Generation Network Infrastructures (NGNI),
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{konrad.campowsky | thomas.magedanz | sebastian.wahle}@fokus.fraunhofer.de

[2] Technische Universität Berlin, Lehrstuhl Architekturen der Vermittlungsknoten (AV),
Franklinstr. 28-29, 10587 Berlin, Germany
{harjoc | tm}@cs.tu-berlin.de

Abstract. Current Future Internet research activities require suitable test and experimental facilities to run large scale experiments. Such testbeds need to be both flexible and robust enough to accommodate different types of experiments including different technologies (e.g. virtualized large scale overlay network testbeds vs. wireless grid testbeds). The very broad and diverse field of Future Internet research and its associated communities require similarly broad and diverse experimental infrastructures that serve the different needs across domains and layers. Therefore, service oriented testbeds that enable cross-layer set-ups (network vs. application space) are important. Furthermore, federation between different testbeds is foreseen to address the cross-domain requirement as single testbeds cannot offer the full range of technologies and mechanisms that are required by the eclectic Future Internet research community. This paper compares different Future Internet control planes in GENI and FIRE and proposes an architecture to address the problem of heterogeneous resource provisioning in a cross-layer and cross-domain fashion. First implementation results are given.

Keywords: Testbeds, Federation, Network Domain Federation, SOA, Management, Provisioning

1 Introduction

The original Internet architecture design and its protocols are from the Seventies. Numerous fixes and extensions have been introduced since then to address a variety of problems such as scalability and security. This led to a highly heterogeneous landscape of different protocols and technologies. Currently, worldwide research activities are under way to investigate alternative solutions and design the architecture of a so-called Future Internet (FI). In order to test and validate their findings, experimental facilities (testbeds) are needed. In addition to the development of innovative foundational architectures, the setup and provisioning of large scale testbeds is considered of major importance in international research. The most prominent examples in this context are the United States NSF programs GENI (Global Environment for Network Innovations) [1] and FIND (Future Internet Design) [2] as well as the European FIRE (Future Internet Research & Experimentation) [3,4] initiative. The focus of GENI is on the design of experimental platforms whereas FIND is mainly addressing foundational concepts and methods for the Future Internet. In the context of GENI, there are currently five competing testbed control frameworks (TIED [5,13], PlanetLab [6], ProtoGENI [7], ORCA [8], ORBIT [9]) under development that are organized in clusters. In the FIRE context, several projects (e.g. Onelab2 [10], Federica [11], PII [15,16]) are contributing to the experimental facility. An in-depth discussion and comparison between the different control framework approaches has been published earlier [12], in the second section we will summarize this briefly to give a short introduction.

Furthermore, the current trend of federation (or more specifically: testbed federation) is driving a number of research activities in the field and influences the design decisions for testbed control frameworks. Federation, combining infrastructural resources and services of more than one independently controlled domain, enhances the utility of testbeds significantly. This is true for the following reasons: access can be given to more resources, increasing the scale of experiments. Furthermore, individual testbeds may include unique infrastructural resources or configuration properties that allow experimenters to execute new kinds of experiments. Finally, because testbeds act as gathering points for experimenters in a given field, combining testbed resources can promote collaboration between very different communities (e.g. Internet community and Telco community) and research groups [13]. Our definition of federation in the context of sharing network resources and services is the following: Federation is a model for the establishment of a large scale and diverse infrastructure for communication technologies, services, and applications and can generally be seen as an interconnection of two or more independent administrative domains for the creation of a richer environment and for the increased multilateral benefits of the users of the individual domains [15]. Testbed federation is currently being investigated and implemented by a number of initiatives to enable network and Future Internet research experiments as well as industrial prototyping and testing. This article analyses current approaches in the field of large scale testbed federation and proposes a control framework architecture that builds upon Service Oriented Architecture principles.

2 Overview of Current FI Provisioning and Control Framework Approaches

Figure 1 gives a high level overview of the main initiatives and the existing frameworks used by them. Please note that the positioning of the entities is not intended to reflect any layering.

The GENI work is organised in so-called spirals where the findings of each spiral are assessed towards the end of a spiral and define the requirements for the next spiral phase. GENI is currently in the first spiral phase [17]. The general high level GENI architecture defines several entities and functions following the Slice-Based Facility

Architecture (SFA) [14]. We introduce them here briefly to allow a comparison with our own approach and ease the understanding of table 1. The most important GENI architecture functions are: slices, components, aggregates, and a clearinghouse.

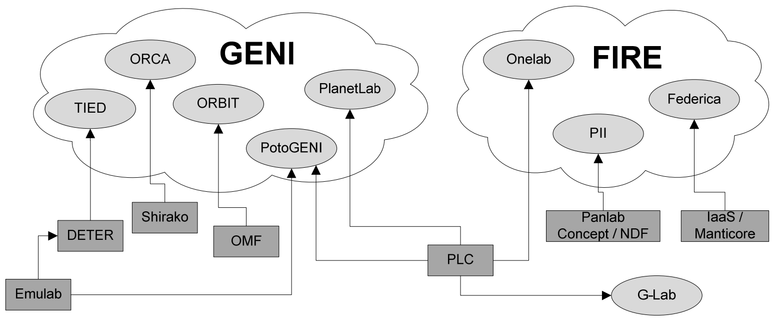

Fig. 1. Overview of the main GENI and FIRE control frameworks and their underlying concepts and existing implementations

Components are the offered resources that are independently owned and operated and can be organized into aggregates, which are groups of resources owned and administered as an ensemble by some organization. Aggregates and its components are available for experiments via a control framework run by a clearinghouse (there will be multiple clearinghouses which will federate). A slice is a set of slivers (a sliver is a part of a specific resource) spanning a set of network components, plus associated users that are allowed to access those slivers for the purpose of executing an experiment. More detailed information on the GENI architecture, its components and its draft control framework can be found in [18,19]. A detailed analysis of the different exiting GENI control framework approaches and current implementations can be found in [12], table 1 summarizes and compares them.

In addition to GENI, there are several European Future Internet control framework research projects and activities organized under the FIRE umbrella as well national initiatives. The FIREworks Specific Support Action [20] coordinates and supports the interworking of testbed activities in Europe. However, no technical specifications are produced by this consortium. Lately, a specific Working Group on modular federation of FIRE Facilities has been founded with the objective of deriving an outline of the architectural principles for a high-level FIRE federation architecture. The aim is not only to address technical but also operational and legal aspects as well as issues related to different business models. It is expected that the group sets the framework (including defined and agreed common services and interfaces/gateways) for the integration of future facility projects submitted under the next FIRE Calls.

Table 1. The GENI and FIRE control frameworks.

	TIED	**ProtoGENI**	**PlanetLab**	**ORCA**	**ORBIT**	**Federica**	**Onelab**	**G-Lab**	**Panlab/PII**
Existing Framework / Concept	DETER, Emulab / DETER Federation Architecture (DFA)	Emulab, PLC / Slice Based Facility Architecture (SFA)	PLC / SFA	Shirako	OMF	IaaS Framework, Manticore	PLC / SFA	PLC	Panlab concept / NDF
Targeted Resources & General Scope	Nodes, links, network capacity, tools (traffic generators, simulators, analyzers) Addresses different communities through federation, rich trust/security models	High-speed backbone on Internet2's wave infrastructure, nodes, links	Nodes, VMs	Resource leasing, BEN resources, dark fiber, reconfigurable fiber switches, DWDM, layers 2 and 3 switch/router, virtual computing clusters at different sites, network tunnels, storage, high-end computing resources	Specific focus on mobile grid testbeds and mobile nodes	NREN and GÉANT2 infra-structure, Gigabit Ethernet circuits, layer 2 and layer 3 switches, PCs	Includes Wireless, DTN, AC, content-based networks	Nodes, VMs, wirless extension planned	Aims to be generic, faces the challenges of high resource heterogeneity
Resource / Experiment Description	Specific focus on experiment description, Canonical Experiment Description Language (CEDL), network simulator (ns), Emulab ptop/vtop format	ProtoGENI Rspec, XML, RELAX NG, switch to GENI Rspec foreseen	PlanetLab Rspec, XML	Exploring resource description beyond current PlanetLab Rspec using and extending network description language (NDL)	XML, specific focus on experiment description including: topology, applications, mapping, are also looking at semantic resource descriptions beyond Rspec (RDF)	Not yet fixed	See PlanetLab	See PlanetLab	Not yet fixed. Probably use of domain specific languages (e.g. NDL for network layer) to support various communities, high level mapping is challenging
Control Plane Message Technology	Web Services, SOAP, XML-RPC	XML-RPC and HTTP(S)	XML-RPC and HTTPS, GENIwrapper module will support final GENI technology	SOAP with WS-Security, XML-RPC	REST, are also looking into a messaging-based framework based on XMPP	Foreseen to move to SOAP Web Services	See PlanetLab	See PlanetLab	Web Services, currently REST, SOAP is under evaluation

3 NDF Applied as FI Testbed Provisioning Concept

The general concept of NDF has been described earlier [21,22]. We review it briefly here to introduce the main entities that are relevant for the CF design discussion. NDF assumes that several administrative domains (network domains that are each represented by an administrative organization) agree to collaborate, forming a federation. The federation itself is represented by an organization (business entity, e.g. federation office) that offers a variety of operational and legal services to the domains and external parties. The domains are interconnected and offer heterogeneous resources as services to external parties or other domains. A central high-level tool allows for combination and interconnection of resources from arbitrary domains in order to offer meaningful infrastructure set-ups and control them remotely. Figure 2 illustrates this concept using PII project [16] entity names and interface descriptions.

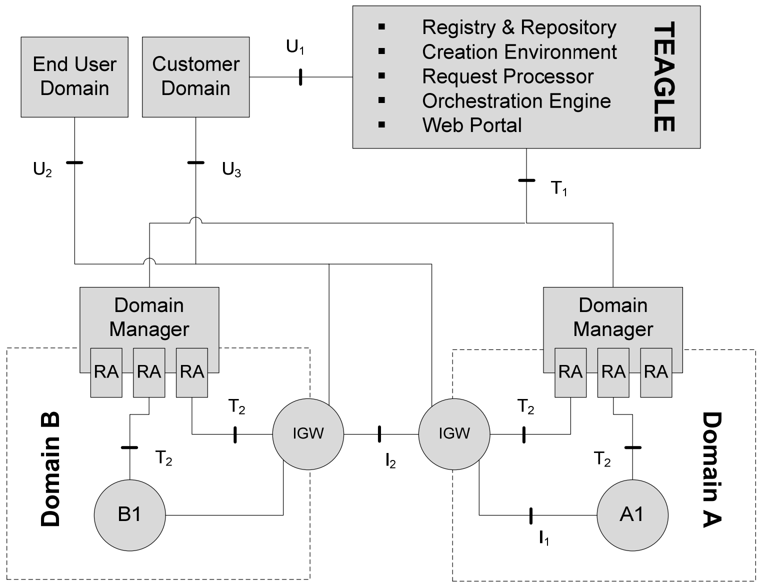

Fig. 2. Making use of the NDF concept – A PII architecture overview.

The resources that are offered by the domains are shown in figure 2 as the circles A1 and B1. The resources may be highly heterogeneous and located in different layers. The concept explicitly does not limit the scope and type of resources. This is what differentiates NDF from other concepts like classical Grid Computing where resources tend to be homogenous or at least somewhat limited in scope. Given the high resource heterogeneity, an abstraction layer is necessary to allow for common management and provisioning routines. This is offered by a Domain Manager (DM) per administrative domain combined with pluggable Resource Adaptors (RA). The RAs can be seen as device drivers that support resource specific types of communication (interface T2 in figure 2). Resources located in different administrative domains are interconnected via Interconnection Gateways (IGW) that allow for connections on different layers (e.g. layer 2/3 VPN). The DM offers generic management operations (such as add, delete, modify) or resource specific service interfaces towards the upper federation logic Teagle on interface T1. T1 is foreseen to be a Web Service interface. Teagle is the central federation tool that combines several functions such as a resource registry, a customer interface (including search functionality and a creation environment), orchestration logic, and more. It can be seen as an extended clearinghouse in GENI terminology.

The NDF control framework is currently implemented (see section 4) and first results show that the main challenge will be to describe the different domain resources in an appropriate level of detail and build a consistent model that allows the upper federation logic to control the distributed resources. Existing description languages and models are usually designed to serve a specific purpose relevant to the respective community (e.g. Network Description Language (NDL) [23] for network descriptions). Combining the different existing approaches on different layers (e.g. network layer vs. application space) and mapping between them, will allow different communities to use natural resource description techniques while enabling higher level federation logic.

At Fraunhofer FOKUS, we investigate and implement this concept based on our numerous in-house testbeds. With our test environments such as the Open SOA Telco Playground [24] and the Future Internet Lab [25] we actively participate in a number of Future Internet research and experimental facility projects such as Onelab, PII, G-Lab [26], 4WARD [28] and others as the basis for experimentally driven and industry near research.

4 Implementation

This section gives an overview of our resource provisioning and control framework based on the NDF concept. In the following subsections we will discuss especially our prototypes for Teagle and the Domain Manager in detail.

4.1 Teagle Prototype Implementation

The Teagle prototype implementation is the central search and composition engine (see figure 2) of the Panlab testbed federation that is based on the NDF concept. It provides a web-based customer interface for browsing the federation’s offerings and the provisioning of Virtual Customer Testbeds (VCT).

A VCT is the sum of all resources and interconnections configured and rented by a specific customer. It is an isolated network where the customer has direct access to the resource and configurations assembled by using Teagle. Each customer operates inside its own VCT and has no access to other VCTs.

Teagle combines several functions that are currently implemented in a centralized manner (although distribution is generally possible but requires more complex trust hierarchy concepts):

- Registry & Repository (user, resources, configurations)
- Creation Environment (setup and configuration of VCTs, this is the VCT designer tool)
- Request Processor (for validating VCT configurations and trigger setup execution)
- Orchestration Engine (for generation of an executable workflow that orchestrates services form different domains to actually instantiate a VCT)
- Web Portal (for exposing search and configuration interfaces, e.g. the VCT designer tool as shown in figure 3)

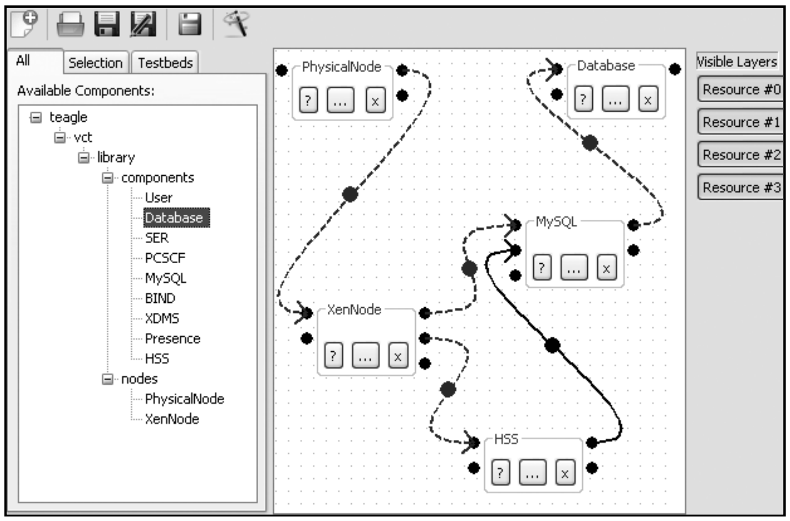

Fig. 3. The VCT designer tool as one of the main Teagle assets.

In the following, we will describe the VCT designer tool part of Teagle in more detail that allows for configuration of testbed setups and topologies. The output of the VCT designer tool (a specified XML document defining the custom VCT configuration) is passed to the Request Processor and then to the Orchestration engine for further processing and execution.

As shown in figure 3, the VCT designer tool combines a selection panel on the left hand side that allows browsing available federation resources and getting information on their functions and availability. Selected items (e.g. HSS, MySQL, etc.) can be placed and interconnected in the workbench. The arrows interconnecting the items have specific semantics. In the setup shown in figure 3, the solid lines represent a protocol interconnection while the dotted lines reflect containment. This means for example that the MySQL server is hosted (contained) by the XenNode that itself is hosted by the physical node. The VCT layout and its associated resource configuration settings can be saved, upon which a XML document is produced. This is the input for the Orchestration Engine that transforms the VCT specification (a list of involved resources, their configuration, and some dependencies) into an executable script that launches the actual VCT instantiation (the provisioning of resources in different domains). Upon execution of the script, Web Service requests are sent to all involved Domain Managers triggering the deployment and configuration of resources according to the VCT specification.

For the development of the VCT designer tool which is a Java implementation we relied on the following technologies and software projects:

- Java SWT - The Standard Widget Toolkit (for the VCT designer tool window)
- SWTCalendar (for availability filtering functions on the selection panel)
- Java XStream (for serialization of objects to XML and back)
- eXist Database (for storing resource descriptions)
- Apache log4j (for logging at different levels like DEBUG, ERROR, INFO, etc.)

The VCT designer tool is under development and has been published to a small tester community that is currently evaluating its features and usability. Our prototype makes a compromise between the simplicity of web-based interfaces (no installation, no maintenance, portability of user profiles) and the power of desktop applications. In its current implementation, the VCT designer runs as a Java Web Start application, and can be launched from the Teagle website [27] as needed[1]. In order to avoid storing user profiles and VCTs on customers' local machines, the Web Start application uses the Teagle registry and repository for storage.

The repository is designed on top of an XML database. The primary reason for this was to allow queries on more than just the tabular structure of SQL. This ability was needed since several registries within the repository have an inherently hierarchical structure (such as our resource model, the resource and VCT registries, or the resource configuration formats). A second benefit that a database with a dynamic structure brings, is the low amount of code changes required when this structure changes. Since the current implementation is still in a prototype status, changes occur regularly. Not having to explicitly restructure the database upon such changes reduces the development time considerably. Introducing a wrapper for the raw interface to the database makes it possible to restrict access to it for components that are not considered secure. Nevertheless, internal components have the right to issue select/update queries directly on the database, using XML query languages (currently XPath [29] and XUpdate [30]).

The mechanism used for synchronizing access to the repository and Teagle components connected to it is the MVC (Model-View-Controller) paradigm. In MVC terms, the repository is the model, the VCT designer tool represents the view and the

[1] The VCT tool is currently only accessible to a small testing group. A public launch is foreseen for the 4th quarter of 2009.

Request Processor is the controller. Updates in the repository trigger notifications that components can register to (as an example, the Request Processor is notified about new VCTs inserted into the repository by the VCT designer tool and takes the steps necessary for provisioning these testbeds via the Orchestration Engine, and pushing the result of the provisioning operation back into the repository).

4.2 Domain Manager Prototype Implementation

This subsection describes our Domain Manager (DM) prototype implementation that has been implemented in Python and Java. The DM is a framework for different resource adaptors (RA) to plug in. DM framework RAs can be implemented in Python or Java following the DM framework requirements for pluggable modules. As mentioned before, RAs are like device drivers that support resource specific communication mechanisms on interface T2 (see figure 2 and 4). Examples are Simple Network Management Protocol (SNMP) or Service Provisioning Markup Language (SPML) based messages as well as command-line interface (CLI) commands. So basically any type of resource can be supported by our DM as long as a RA can be implemented and the configuration options can be described and modeled so that the VCT designer tool can handle them. This approach allows us to manage heterogeneous resources that support a variety of different communication mechanisms, reside on different layers (cross-layer), and inside different administrative domains (cross-domain).

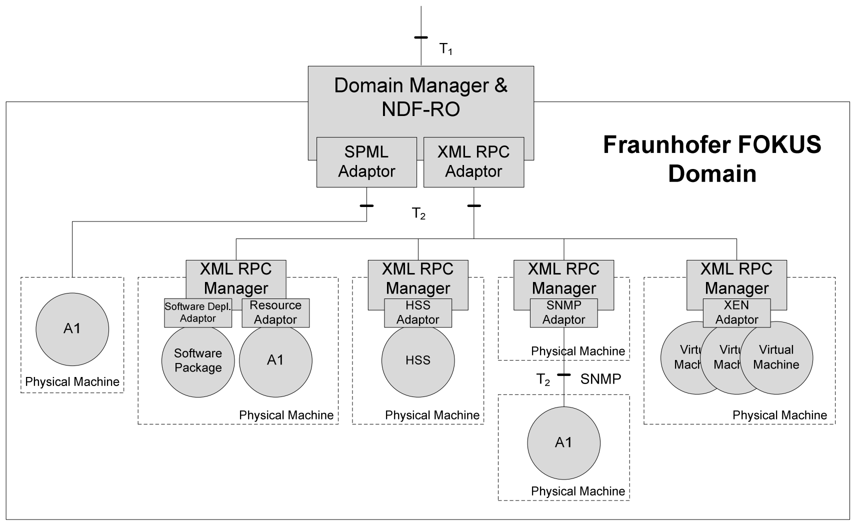

Fig. 4. Overview of the Fraunhofer FOKUS Domain, the associated Domain Manager, and a selection of domain resources.

As shown by figure 4, the DM controls several resources in its domain (e.g. virtual machines, a Home Subscriber Server (HSS) and other arbitrary resources A1). This is possible through several resource adaptors (e.g. XEN [31] Adaptor, SNMP Adaptor, etc.). Through its highly modular structure, the DM architecture supports multiple resource provisioning schemata and languages (e.g. SPML, XML RPC). This flexibility helps to support a variety of resources and their native communication mechanisms through yet another abstraction layer. For example it is possible to instantiate several virtual machines (using XEN) on a physical machine and deploy software resource adaptors on the virtual machines themselves in order to control both the container and the actual software resource residing inside the virtual node. On the other hand, resources that already natively support a provisioning schema, like for example SPML, can be controlled directly without artificially inflating the communication mechanism. We call this concept of both pluggable resource adaptors (SNMP, CLI, etc.) as well as pluggable provisioning schemata (SPML, XML-RPC, etc.) Network Domain Federation Remote Objects (NDF-RO).

On interface T1, we currently rely on a REST (Representational State Transfer) implementation. Management operations (add, delete, modify resources) supported by the individual resources are exposed as REST services. We are currently also working on a SOAP (Simple Object Access Protocol) implementation of T1 to allow for more complex resource management operations and allow for pure SOA style service orchestration and high level resource abstractions.

For the implementation of our DM and the NDF-RO concept prototype we rely on the following technologies and software projects:

- Java and Python (to help RA implementers and allow them to choose their favorite language for implementing DM framework RAs)
- Apache log4j (for logging at different levels like DEBUG, ERROR, INFO, etc.)
- JDOM (for manipulating, and outputting XML data from Java code)
- Apache XML-RPC Facilities for Java
- Apache Axis2 (for T1 SOAP interface)
- Java Tomcat Servlet API (for HTTP access to DM)
- Apache HttpComponents and Python Standard Library (for REST based T1)

Next steps in the DM and NDF-RO architecture development will be the realization of an advanced software repository to allow many different types of software images (and different versions of those) to be accessible via our control framework. Also, the further diversification of resources (more hardware devices, sensors, wireless nodes, etc.) will be crucial to establish this concept with a real benefit for the Future Internet research community and their very different needs.

5 Conclusion

We observe that at the current stage it is unclear how new concepts and ideas developed by the Future Internet research community can be integrated into productive systems while preserving current operator infrastructure investments. From our perspective service oriented experimental facilities are required that integrate technologies and concepts across various domains and layers and actively incorporate the industry.

The analysis of the current control frameworks shows that most of them are limited to specific types of resources to serve a specific community. We believe that the Future Internet will have to deal with a variety of different technologies and devices on various layers including mobility. Therefore, it is crucial to design Future Internet experimental facilities and an associated control framework and tools that abstract from the underlying heterogeneity and integrate different facilities and resources across domains and layers. With NDF and its modular design, we propose a concept and architecture to achieve this in a service oriented fashion. Our current implementations of a Virtual Customer Testbed (VCT) designer tool and the underlying resource-near execution environment (Domain Manager & NDF-RO) show that it is possible to accommodate a variety of resources and support their management in a generic fashion. However, we are still facing challenging research tasks in the field of harmonizing different resource description languages and techniques. Also, the support of different research communities with different requirements and testing and development habits will require serious efforts as well as constant adaptation to changing demands.

Acknowledgments.
Parts of this paper are based on work and ideas generated in the course of the project PII [15,16], which receives funding from the European Commission's Seventh Framework Programme (FP7).

References

1. National Science Foundation, GENI website: http://www.geni.net
2. National Science Foundation, FIND website: http://www.nets-find.net
3. European Commission, FIRE website: http://cordis.europa.eu/fp7/ict/fire
4. Gavras, A., Karila, A., Fdida, S., May, M., and Potts, M. 2007. Future internet research and experimentation: the FIRE initiative. SIGCOMM Comput. Commun. Rev. 37, 3, 89-92 (2007)
5. Faber, T., Wroclawski, J., Lahey, K.: A DETER Federation Architecture, DETER Community Workshop on Cyber-Security and Test (2007)
6. Peterson, L. and Roscoe, T: The Design Principles of PlanetLab. SIGOPS Oper. Syst. Rev. 40, 1, 11-16 (2006)
7. GENI Project Office, ProtoGENI Control Framework Overview, GENI-SE-CF-PGO-01.4 (2009)

8. Chase, J et al.: Beyond Virtual Data Centers: Toward an Open Resource Control Architecture, Selected Papers from the International Conference on the Virtual Computing Initiative (ACM Digital Library) (2007)
9. Ott, M et al.: ORBIT Testbed Software Architecture: Supporting Experiments as a Service, Proceedings of IEEE Tridentcom 2005 (2005)
10. OneLab project website, http://www.onelab.eu/
11. Campanella, M.: The FEDERICA Project - A federated infrastructure for Future Internet research, EURESCOM mess@ge, issue 2/2008 (2008)
12. Thomas Magedanz and Sebastian Wahle. Control Framework Design for Future Internet Testbeds. e & i Elektrotechnik und Informationstechnik, 07/08, August 2009. ISSN: 0932-383X (print) ISSN: 1613-7620 (online), accepted for publication in 2009 (August).
13. Faber, T., Wroclawski, J.: A Federated Experiment Environment for Emulab-based Testbeds, ICST (2009)
14. Peterson, L. et al.: Slice-Based Facility Architecture, Draft Version 1.04, http://svn.planet-lab.org/attachment/wiki/GeniWrapper/sfa.pdf (2009)
15. Anastasius Gavras, Halid Hrasnica, Sebastian Wahle, David Lozano, Denis Mischler, and Spyros Denazis. Towards the Future Internet - A European Research Perspective, chapter Control of Resources in Pan-European Testbed Federation, pages 67 - 78. IOS Press, ISBN 978-1-60750-007-0 (2009)
16. Website of Panlab and PII European projects, supported by the European Commission in its both framework programmes FP6 (2001-2006) and FP7 (2007-2013): http://www.panlab.net
17. GENI Project Office, GENI Spiral 1 Overview, Document ID: GENIFAC-PRO-S1-OV-1.12, September 29, 2008, http://www.geni.net/docs/GENIS1Ovrvw092908.pdf
18. GENI Project Office, GENI System Overview, Document ID: GENISE-SY-SO-02.0, September 29, 2008, http://www.geni.net/docs/GENISysOvrvw092908.pdf
19. GENI Project Office, GENI Control Framework Architecture, Document ID: GENI-ARCH-CP-01.4, October 20, 2008, http://groups.geni.net/geni/attachment/wiki/GeniControlFrameworkArchitecture/102008_GENI-ARCH-CP-01.4.pdf
20. FIREworks, FIREworks SSA consortium, website http://www.ict-fireworks.eu
21. Wahle, S. et al.: Network Domain Federation - Infrastructure for Federated Testbeds. 2008 NEM Summit - Towards Future Media Internet. Saint-Malo, France: Eurescom GmbH: 179 – 184 (2008)
22. Thomas Magedanz, Florian Schreiner, and Sebastian Wahle. Service-Oriented Testbed Infrastructures and Cross-Domain Federation for Future Internet Research. In 2009 IFIP/IEEE International Symposium on Integrated Network Management Proceedings, New York, USA, June 2009. IEEE. Accepted for publication in 2009 (June).
23. University of Amsterdam, NDL website: http://www.science.uva.nl/research/sne/ndl
24. Fraunhofer FOKUS, Open SOA Telco Playground website: www.opensoaplayground.org
25. Fraunhofer FOKUS, Future Internet Lab website: www.fokus.fraunhofer.de/go/fi-lab
26. Tran-Gia, P: G-Lab Phase 1, Whitepaper, http://www.german-lab.de (2008)
27. Teagle website: http://www.fire-teagle.org/
28. 4WARD project website: http://www.4ward-project.eu/
29. XML Path Language (XPath) 2.0, W3C Recommendation (2007)
30. XUpdate - XML Update Language Working Group website: http://xmldb-org.sourceforge.net/xupdate/index.html
31. Xen-Projekt-website: http://xen.org/