29-05-2012

Deliverable DJ1.4.2: Virtualisation Services and Framework – Study

Deliverable DJ1.4.2

Contractual Date:	31-03-2012
Actual Date:	29-05-2012
Grant Agreement No.:	238875
Activity:	JRA1
Task Item:	Task 4
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	JANET/University of Essex
Document Code:	GN3-12-123
Authors:	B. Belter (PSNC), M. Campanella (GARR), F. Farina (GARR), J. Garcia-Espin (I2CAT), J. Jofre (I2CAT), P. Kaufman (DFN), Radek Krzywania (PSNC) ,L. Lechert (PSNC), F. Loui (RENATER), R. Nejabati (University of Essex), V. Reijs (HEANET), C. Tziouvaras (GRNET), T. Vlachogiannis (University of Essex), D. Wilson (HEANET)

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 238875 (GÉANT).

Abstract

This deliverable updates the results of a comprehensive comparative study of existing infrastructure virtualisation technologies and frameworks carried out in previous deliverable (DJ1.4.1). It also presents the results of a drawback analysis of virtualisation deployment for NRENs and GÉANT. In addition, the deliverable defines a multi-layer, multi-domain virtualisation service for GÉANT, GENUS, proposing an architecture as well as an approach for its implementation within GÉANT and associated NREN infrastructures. Finally, it provides details of GENUS prototype design and its proof-of-concept implementation on the GENUS testbed.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This document aims to investigate potential uses and benefits of infrastructure virtualisation services to provide guidance for the GÉANT and NREN communities. It proposes a multi-layer, multi-domain and multi-technology virtualisation architecture suitable for NREN requirements based on tools and software that have already been developed or are currently under development within the European research community.

The deliverable first updates the comprehensive comparative study of recent and existing major activities, research projects and technologies addressing infrastructure virtualisation that was begun in "Deliverable DJ1.4.1: Virtualisation Services and Framework Study" [GN3-DJ1.4.1]. The projects considered include European projects (FEDERICA, MANTYCHORE, Phosphorus, 4WARD, GEYSERS, NOVI, OFELIA), US projects (GENI, PlanetLab/VINI/OneLab), a Japanese project (AKARI) and two commercial cloud projects (Amazon virtualisation and Google App engine). All the projects include infrastructure virtualisation at national and/or international level and some of them involve National Research and Education Networks (NRENs) and international connectivity. The study tries to provide a consistently structured assessment of the different projects, addressing the following points:

- Overview of the project and its objective.
- A definition of infrastructure virtualisation as understood by the project as well as an architectural overview of its virtualisation approach.
- User community.
- Overview of existing features and implementation of virtualisation for Layer 1, Layer 2, Layer 3 and computing resources.
- Multi-domain support offered by the virtualisation technology.
- Testbed implementation and availability.
- Current status and roadmap.

The results of the study conclude that the European research community, helped by the drive and commitment of the NRENs, has achieved significant progress on infrastructure virtualisation technologies through projects such as GEYSERS, MANTYCHORE, NOVI and OFELIA. These projects are complementary and, combined together, can provide virtualisation of Layer 1, Layer 2 and Layer 3 networks as well as computing resources. The involvement of GN3 JRA1 Task 4 participants in these projects means the Task is well placed to build on their developments and achievements, leveraging the first-hand knowledge and experience gained in defining its proposal for GÉANT virtualisation services, at the same time providing the capability to incorporate the outcome of any future relevant projects and frameworks.

This deliverable also presents the results of a drawback analysis of virtualisation deployment for NRENs and GÉANT, considering technical, service and business issues, and assessing the probability and severity of each potential drawback. The analysis concludes that there are no major issues with regard to the technical features required for the hardware and software to provide the necessary capabilities for virtualisation. However, apart from these purely technical aspects, somewhat larger problems still exist, especially with regard to the operational environment, the maturity of solutions and the area of security.

Many of the virtualisation technologies resulting from the projects mentioned above are still in their research and development stage. It is therefore not realistic to propose a specific solution to the NREN and GÉANT community. This report does not aim to promote a specific solution or framework for the GÉANT virtualisation service. Instead, it proposes a solution for integrating and interworking existing virtualisation mechanisms and solutions at different layers, leaving the choice of suitable virtualisation technologies to individual NRENs, while enabling them to offer multi-domain, multi-layer and multi-technology virtualisation service.

The deliverable defines a multi-layer, multi-domain infrastructure virtualisation service for GÉANT which is called GENUS (GÉaNt virtUalisation Service). It proposes an architecture as well as an approach for GENUS implementation based on a combination of solutions and tools provided by relevant EU projects such as OFELIA and MANTYCHORE as well as the GÉANT Bandwidth on Demand service. Without reinventing the wheel, the proposition is to integrate and orchestrate existing Layer 1, Layer 2, Layer 3 and computing virtualisation tools using the GENUS platform.

The deliverable also outlines the software design and implementation of the GENUS prototype. The prototype includes capability for multi-domain Layer 1, Layer 2, Layer 3 and computing virtualisation as well as support for the AutoBAHN bandwidth-on-demand provisioning tool. It also includes a web-based user interface as well as a set of complex methods for virtual infrastructure composition.

In addition, the deliverable discusses all the relevant issues for operational support and service of a virtualised infrastructure. It introduces the required functionality as well as the business and management aspects of a virtual infrastructure from both operator and user point of view by introducing the concept of a Virtualised Operations Support Service (VOSS).

Finally, the deliverable describes the GENUS multi-domain heterogeneous testbed and verification platform. The prototype demonstration and proof-of-concept implementation will be carried out using existing resources within Task 4 participants' facilities. The GENUS testbed comprises two virtualisation frameworks, i.e. MANTYCHORE (IP virtualisation) and OFELIA (computing, Layer 2 and Layer 1 virtualisation based on OpenFlow), with support for GÉANT's AutoBAHN tool all installed over four local testbeds interconnected by GÉANT and FEDERICA. A final set of results will be documented in a white paper, due to be available in March 2013.

1 Introduction

Current developments and technical enhancements of transport networks' technologies, network management and control planes, multi-core processing, cloud computing, data repositories and energy efficiency, are driving profound transformations of NRENs' (National Research and Education Networks) network infrastructures and their users' capabilities. These technological advances are driving the emergence of ever more demanding high-performance and network-based applications with strict IT (e.g. computing and data repositories) and network resource requirements. Examples of these applications include: ultra-high definition remote visualisation and networked high-performance supercomputing infrastructure. These types of applications often require their own dedicated network and IT resources tailored to their strict computing and network resource requirements. As these types of collaborative and network-based applications evolve, addressing the needs of a wide range of users in the NREN community, it is not feasible (for scalability reasons, among others) to set up and configure dedicated network and computing resources for each application type or category. Consequently, NRENs need to deploy an infrastructure management mechanism able to support all application types optimally, each with their own access, network and IT resource usage patterns. Any solution providing such an infrastructure management mechanism has to address the following challenges:

- Increase in the number of users/applications and rapid increase in available bandwidth for users beyond 1 Gbit/s.
- Emergence of new scientific applications requiring 10G or even 100G connectivity e.g. the Large Hadron Collider (LHC) and radio astronomy.
- Partitioning of physical network and IT infrastructures for providing secure and isolated application specific infrastructure.
- Migration towards a full range and large-scale convergence of IT and network services.
- Energy-efficiency in networking and computing.

A key issue in addressing these challenges is efficient network and computing resource utilisation and sharing within the current and future NREN infrastructure.

This deliverable aims to investigate potential uses of virtualisation to address the challenges listed above and provide guidance to the GÉANT community. In the context of network and computing infrastructure, virtualisation is the creation of a virtual version of a physical resource (e.g. network, router, switch, optical device or computing server), based on an abstract model of that resource and often achieved by partitioning (slicing) and/or aggregation. A virtual infrastructure is a set of virtual resources interconnected together and managed by a single administrative entity. The deliverable proposes a multi-layer virtualisation architecture suitable for NREN requirements based on tools and software that have already been developed or are currently under development within the European research community.

GN3 Joint Research Activity 1 Future Network, Task 4 Current and Potential Uses of Virtualisation (JRA1 T4) and, consequently, this deliverable investigate the application of virtualisation technology for the GÉANT community within the framework of Infrastructure as a Service (IaaS) [IaaS]. IaaS is a promising paradigm that enables NRENs to provide infrastructure resources such as routers, switches, optical devices, Internet Protocol (IP) networks, and computing servers as a service to their user communities. It comprises a set of software and tools that allows virtualisation of infrastructure by means of partitioning (slicing) and/or aggregation of infrastructure resources (i.e. network elements and computing resource). Resource virtualisation is an effective method for sharing infrastructure resource among users and applications efficiently and therefore its immediate benefit for NRENs is to increase resource utilisation efficiency. Virtualisation can also potentially enable NRENs to offer remote access and control of virtual infrastructure elements (slices of real physical elements) to their user organisations through web services. By using virtualisation services, users can control their own virtual infrastructure. This provides an effective mechanism for secure and isolated application-specific virtual infrastructures to share physical infrastructure. Furthermore, virtualisation can potentially provide a new level of flexibility to the NRENs, as their infrastructure can scale up or down following user/application requirements, thereby minimising the cost of operating the infrastructure (both the capital and the operational expenditures).

This deliverable is the second and final report on a comparative study of existing virtualisation technologies and frameworks; the initial report, “Deliverable DJ1.4.1: Virtualisation Services and Framework Study” [GN3-DJ1.4.1], was published in May 2010. It updates the previous findings, adds further examples of virtualisation technologies and frameworks, and has a new section covering a drawback analysis of infrastructure virtualisation for NRENs and GÉANT. It also includes a proposal and proof-of-concept implementation for a virtualisation service in GÉANT. To this end the deliverable has been organised as follows:

- Chapter 2 updates the results of a detailed comparative study of the main recent and current projects and initiatives addressing virtualisation technology.
- Chapter 3 reports on a drawback analysis of infrastructure virtualisation.
- Chapter 4 outlines a proposed GÉANT virtualisation service, GENUS, which is a multi-layer and multi-domain infrastructure virtualisation mechanism based on a combination of solutions and tools provided by relevant EU projects.
- Chapter 5 discusses issues relevant to the operation, management and support of a virtualised infrastructure, particularly as addressed by GENUS’ Virtualised Operations Support Service (VOSS).
- Chapter 6 describes the software design of GENUS and the proof-of-concept implementation.
- Chapter 7 describes the GENUS multi-layer and multi-domain virtualisation testbed and virtualisation verification platform for future improvement and investigation of issues relevant to GENUS and the GÉANT virtualisation service.
- Finally, Chapter 8 provides an overall assessment of JRA1 Task 4’s work on virtualisation technologies to date.

2 Overview of Existing Virtualisation Technologies and their Usage

2.1 Introduction

This chapter updates the comprehensive overview of recent and existing major activities, research projects and technologies addressing infrastructure virtualisation that was first documented in DJ1.4.1. Where the information is unchanged from the initial report, it is not duplicated here; instead, a reference to the appropriate section in DJ1.4.1 is provided. For convenience, however, the summary comparison table (Section 2.14) covers *all* projects and initiatives. Table 2.1 below shows the projects considered, giving their type (European, US, Japanese or commercial cloud), and the status and location of the information. All the projects include infrastructure virtualisation at national and/or international level and some of them involve National Research and Education Networks (NRENs) and international connectivity. (Although there are other European virtualisation projects, e.g. the computing virtualisation projects OpenNEBULA and STRATUS lab, and other commercial network virtualisation products, e.g. Tail-f's Network Configuration Server (NCS), these have not been considered because they do not address infrastructure virtualisation as defined within this document, where an infrastructure (network + computing) is sliced by means of virtualisation and control of the slice is given to its users or virtual infrastructure owner.)

Project	Type	Status	Document / Section
FEDERICA	European	Information unchanged	DJ1.4.1 Section 2.2
MANTICORE/MANTYCHORE	European	Updated and/or new	This document Section 2.3
Phosphorus	European	Information unchanged	DJ1.4.1 Section 2.4
4WARD	European	Information unchanged	DJ1.4.1 Section 2.5
GENI	US	Updated and/or new	This document Section 2.6
PlanetLab/VINI/OneLab	US	Information unchanged	DJ1.4.1 Section 2.7
AKARI	Japanese	Updated and/or new	This document Section 2.8
GEYSERS	European	New	This document Section 2.9
NOVI	European	New	This document Section 2.10

Project	Type	Status	Document / Section
OFELIA	European	New	This document Section 2.11
Google App Engine	Commercial cloud	Updated and/or new	This document Section 2.12
Amazon Virtualisation	Commercial cloud	Information unchanged	DJ1.4.1 Section 2.9.1

Table 2.1: Status and location of virtualisation projects' information

The review of each project has been organised to cover:

1. Introduction – an overview of the project and its objective.
2. Architecture overview – a definition of infrastructure virtualisation as understood by the project as well as an architectural overview of its virtualisation approach.
3. User community – a description of the user group(s) at which the project is aimed.
4. Mechanisms for providing virtualisation – an overview of existing features and implementation of virtualisation for Layer 1, Layer 2, Layer 3 and computing resources.
5. Multi-domain support – a statement of whether the virtualisation technology can be applied in a multi-domain environment.
6. Testbed implementation and availability – a description of the virtualisation testbed and test scenario, if they exist.
7. Current status and roadmap – roadmap and future plans with respect to virtualisation.
8. References – details of sources cited in the overview (these are also given in the References section at the end of the document on page 68.)

Finally, this section concludes with a comparison table summarising the virtualisation capability and features of all the projects reviewed.

2.2 FEDERICA

The information about FEDERICA is unchanged. Please refer to [GN3-DJ1.4.1] Section 2.2.

2.3 MANTICORE/MANTYCHORE

Much of the information in this section is taken from the MANTYCHORE "Description of Work" [MANTYCHORE-DoW].

2.3.1 Introduction

In 2006, MANTICORE I's main objective was to implement a proof of concept for the IP Network as a Service (IPNaaS) paradigm, which was successfully demonstrated. A privately funded consortium, MANTICORE II was initiated in 2008 to implement the abovementioned paradigm as a robust tool. At the end of MANTICORE II, three NRENs performed a pilot trial.

In 2010, the MANTYCHORE project started, funded by the FP7 programme. The goal is to provide IPNaaS to three end-user communities: eHealth, Media and Grid/Cloud computing. MANTYCHORE is also intended to exploit the Infrastructure as a Service paradigm to enable NRENs and other e-infrastructure providers to enhance their service portfolio by building and deploying the software and tools to provide IP Networks as a Service to virtual research communities. Another important objective is to improve and expand the services provided by integrating the results of MANTICORE II with solutions based on the optical IaaS Framework [Argia] and Ethernet / Multi-Protocol Label Switching (MPLS), so that offer the possibility of providing integrated services to level 1-3 for the research community.

IP Network as a Service (IP Network Service) is a key enabler of the flexible and stable e- Infrastructures of the future. Today, a myriad of tool prototypes for providing point-to-point links to researchers have been developed. These tools, while providing high-bandwidth pipes to researchers, only address one side of the problem. Researchers who want to create a virtual community to address scientific problems are still connected to each institution’s networks, and it is a hard problem to connect them directly with high-bandwidth pipes because it causes a number of issues such as security or routing integrity. One of the ways of efficiently solving this problem is to create a logically separated IP network (on top of the high-bandwidth pipes), or to use separate instances of virtualised routers, or a combination of both, and to dedicate it to the virtual research community. In order to maximise the flexibility and convenience of this IP Network Service, the users of the virtual community should be able to modify the characteristics of their IP network (such as the addressing, dynamic routing protocols, routing policies, quality of service and so on) by themselves.

The IP Network Service follows the IaaS paradigm, consisting of offering remote access and control of infrastructure elements to third-party organisations through software web services. By using IaaS services these organisations can control the remote infrastructure as if they owned it and be billed either per use or based on a monthly fee, promoting the re-use of existing infrastructure and avoiding the purchase of new devices on the provider and customer sites.

In order to improve the IaaS service, some alternative but very interesting topics will be researched. An infrastructure resource marketplace and the use of renewable energy sources to power e-infrastructures will be developed and included in MANTYCHORE software, enriching both the user community and the roadmap of the MANTYCHORE project

2.3.2 Architecture overview

As mentioned in Section 2.3.1, MANTYCHORE follows the IaaS paradigm, enabling NRENs and other e-Infrastructure providers to enhance their service portfolio by building and deploying software and tools to provide infrastructure resources (such as routers, switches, optical devices, and IP networks) as a service to virtual research communities. Three user roles can be identified in the IaaS scenario:

- **Infrastructure Provider:** The infrastructure owner. This user can assign permissions to the infrastructure resources he owns so that external users can control it. Infrastructure instances can be either physical (e.g. a physical router) or virtual (e.g. a logical router). In the MANTYCHORE case, NRENs are the Infrastructure Providers, offering their infrastructure to user communities.

- **Service Provider or Virtual Operator:** This user can harvest infrastructure instances from one or more Infrastructure Providers and integrate them into his management domain (e.g. he can integrate several routers into an IP network). He can also act as an Infrastructure Provider and reassign the permissions on "his" infrastructure instances so that other Service Providers can control them (it is a recursive process). He normally uses the infrastructure instances of his domain to provide some kind of service to end users (e.g. an IP Network service). In the MANTYCHORE scenario, an international community of researchers could create a virtual organisation with their own dedicated IP network (built using resources from different NRENs). One partner of this international research community would adopt the role of the "Service Provider" – typically the leader of the testbed Work Package in a European project, for example.

- **End User:** Uses the services offered by the Virtual Operator. These users belong to a virtual community that receives several infrastructure resources and creates one or more IP networks out of them. Users are empowered to change some attributes of the IP network service (such as internal routing, IP addressing, peering, creating circuits between end points, firewalls), but would not be able to modify the number of resources in the network. In any case, it would be their Virtual Operator who controls the permissions of each individual user (hence the definition of different user profiles is possible).

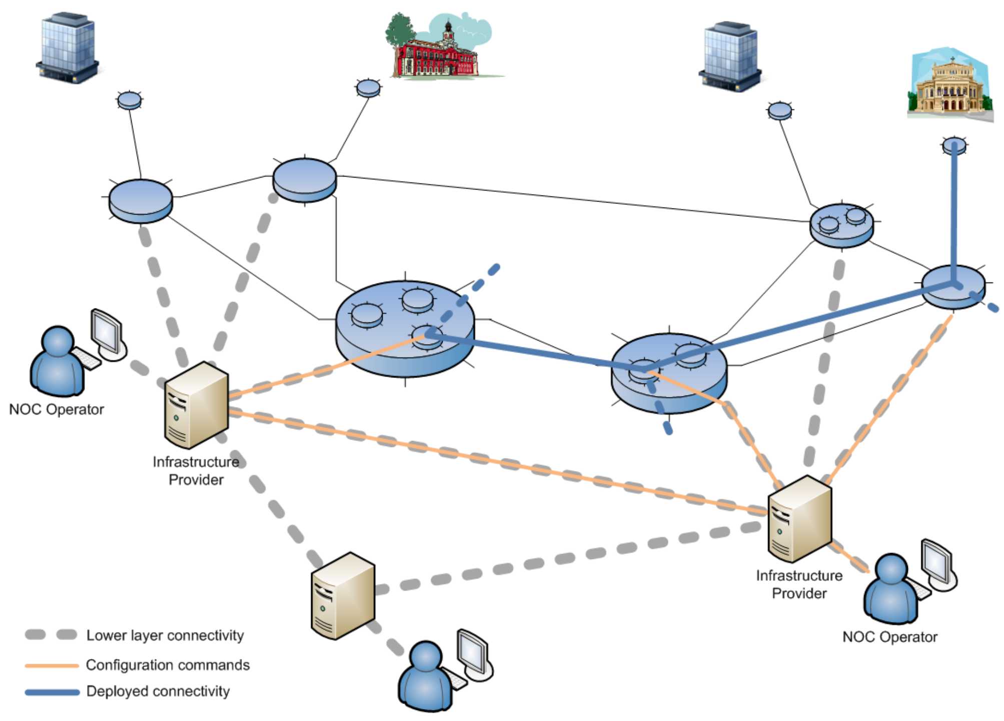


Figure 2.1: MANTYCHORE architecture

Additionally, a marketplace provides a single venue that facilitates the sharing of information about resources and services between providers and customers. It provides an interface through which consumers are able to access the discovered resources from their respective providers. The MANTYCHORE marketplace represents a virtual resource pool that provides a unified platform in which multiple infrastructure providers can advertise their network resources and characteristics to be discovered by potential consumers of the resource. Thus, the marketplace involves three types of entities:

1. The customers that use the resources. These customers may be end users, service providers or other providers who wish to extend their point of presence.
2. The infrastructure providers that provide information about the state of their underlying infrastructure to satisfy the demands of customers.
3. The matchmaking entity that is used to look up and locate relevant resources as requested by the customer. The matchmaking entity mediates among the providers and the customer and uses a matching algorithm that parses requests into queries, evaluates the queries against the resources in the marketplace repository and returns the relevant resources. These algorithms are implemented in a generic manner using Quality of Service (QoS) parameters suitable to Layer 3, 2 and 1.

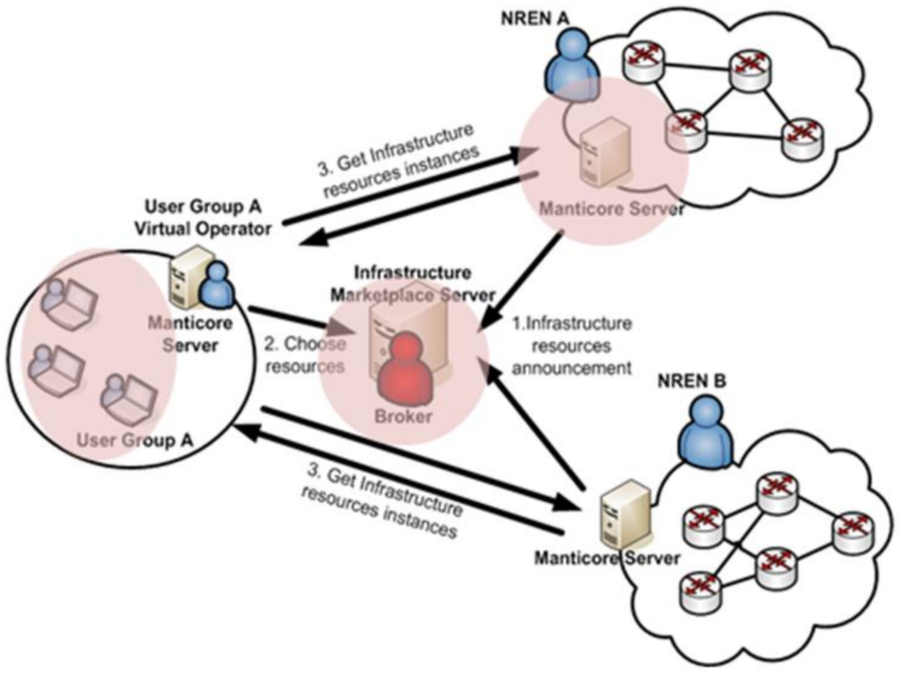

NREN A
3. Get Infrastructure resources instances
User Group A Virtual Operator
Manticore Server
Infrastructure Marketplace Server
Manticore Server
2. Choose resources
Broker
1.Infrastructure resources announcement
NREN B
User Group A
3. Get Infrastructure resources instances
Manticore Server

Figure 2.2: MANTYCHORE marketplace

It is important to note that an adapter needs to be developed for each equipment vendor. Currently, there is support for routers from Juniper and Cisco as well as software routers with Linux running Quagga are in the roadmap.

2.3.3 User community

The initial MANTYCHORE user community is formed of three research user groups, where each user group uses the MANTYCHORE services individually for its own interests. These three user groups include the Danish HDN (Health Data Network), the British UHDM (Ultra High Definition Media) group, and the Irish Grid network.

As MANTYCHORE deployment is in a pre-operational phase, albeit with real users, feedback will be collected from the users to improve the MANTYCHORE services and correct any bugs that may appear. It is not a pilot phase to correct some bugs, it is an evaluation that determines whether the MANTYCHORE services are useful for each particular research community. When the pre-operational phase has been successfully completed, the service can be rolled out on an operational level to a larger community.

2.3.4 Mechanisms for providing virtualisation

2.3.4.1 *Implementation of virtualisation on Layer 3*

The MANTYCHORE suite includes a set of features for:

- Configuration of virtual networks.
- Configuration of physical interfaces.
- Support of routing protocols, both internal (Routing Information Protocol (RIP), Open Shortest Path First (OSPF)) and external (Border Gateway Protocol (BGP)).
- Support of QoS and firewall services.
- Creation, modification and deletion of virtual resources: logical interfaces, logical routers.
- Support of IPv6. It allows the configuration of IPv6 in interfaces, routing protocols, networks.

2.3.4.2 *Implementation of virtualisation on Layer 2*

Users will be able to obtain permissions over Ethernet and MPLS (Layer 2.5) switches, and to configure different services. For this aspect, MANTYCHORE will integrate the Ether project [Ether] and its capabilities for the management of Ethernet and MPLS resources.

2.3.4.3 *Implementation of virtualisation on Layer 1*

Users will be able to obtain permissions over optical devices such as optical switches, and to configure some important properties of the device's cards and ports. For this aspect, MANTYCHORE will integrate the Argia framework [Argia], which provides complete control of optical resources.

2.3.4.4 *Implementation of computing virtualisation*

MANTYCHORE does not define or include virtualised computing infrastructure. Its scope is the connectivity between such resources, and providing an effective method for describing and implementing the connectivity that they require.

2.3.4.5 *Management of virtualised infrastructure*

In general, it is planned that MANTYCHORE will take over management of the physical routers that provide the virtual infrastructure. The routers are configured by the MANTYCHORE server using the NETCONF protocol over Secure Shell (SSH). There may be some flexibility with regard to manual configuration, however.

Once MANTYCHORE is able to manage a physical router, it manages the setup, configuration and deletion of logical routers within the physical device. Normal router management systems, such as Simple Network Management Protocol (SNMP)-based tools like Multi-Router Traffic Grapher (MRTG), can be configured to monitor the infrastructure.

The operator retains the ability to manage the infrastructure, even when they delegate control over a logical router to the user. This ensures that the operator continues to maintain a full picture, and control where necessary, of the overall infrastructure, while day-to-day management is delegated (within particular boundaries) to the user.

2.3.4.6 *Control of virtualised infrastructure*

Control of the infrastructure in MANTYCHORE is delegated by the infrastructure operator to the user, perhaps through a network operations centre or a customer’s IT department. The objective is to allow the user to perform the day-to-day setup and management of their logical network, within the boundaries agreed to and set by their provider.

As a result, where MANTYCHORE manages a set of physical routers with particular facilities, the operator of the physical router will set up logical routers and delegate access to them, with particular interfaces and protocols, to their customer. This access may be further delegated to an end user. The end user can then set up the network of their choice, within the boundaries delegated to them.

2.3.4.7 *Implementation of user interface*

The process of setting up and delegating routers from the operator to the user is accomplished by means of the Graphical User Interface (GUI). This will work in a web browser, accessing a backend server which itself will configure the routers. The operator needs to set up the physical routers for management, and can then set up logical routers and delegate access to them to a user.

Once the user has logical routers delegated to them, they also configure the routers by means of the MANTYCHORE GUI. Links between the routers are created, IP addresses defined and routing protocols as needed are set up. While this was a relatively manual process in MANTICORE II, the MANTYCHORE project plans a process that is much more intuitive, automating best-practice networking in such a way that users should be able to set up their networks, to the level of detail they are comfortable with, without requiring specialist networking knowledge. Currently, the user interface is the same as that used in MANTICORE; a screenshot of the IP Network editor is shown in Figure 2.3.

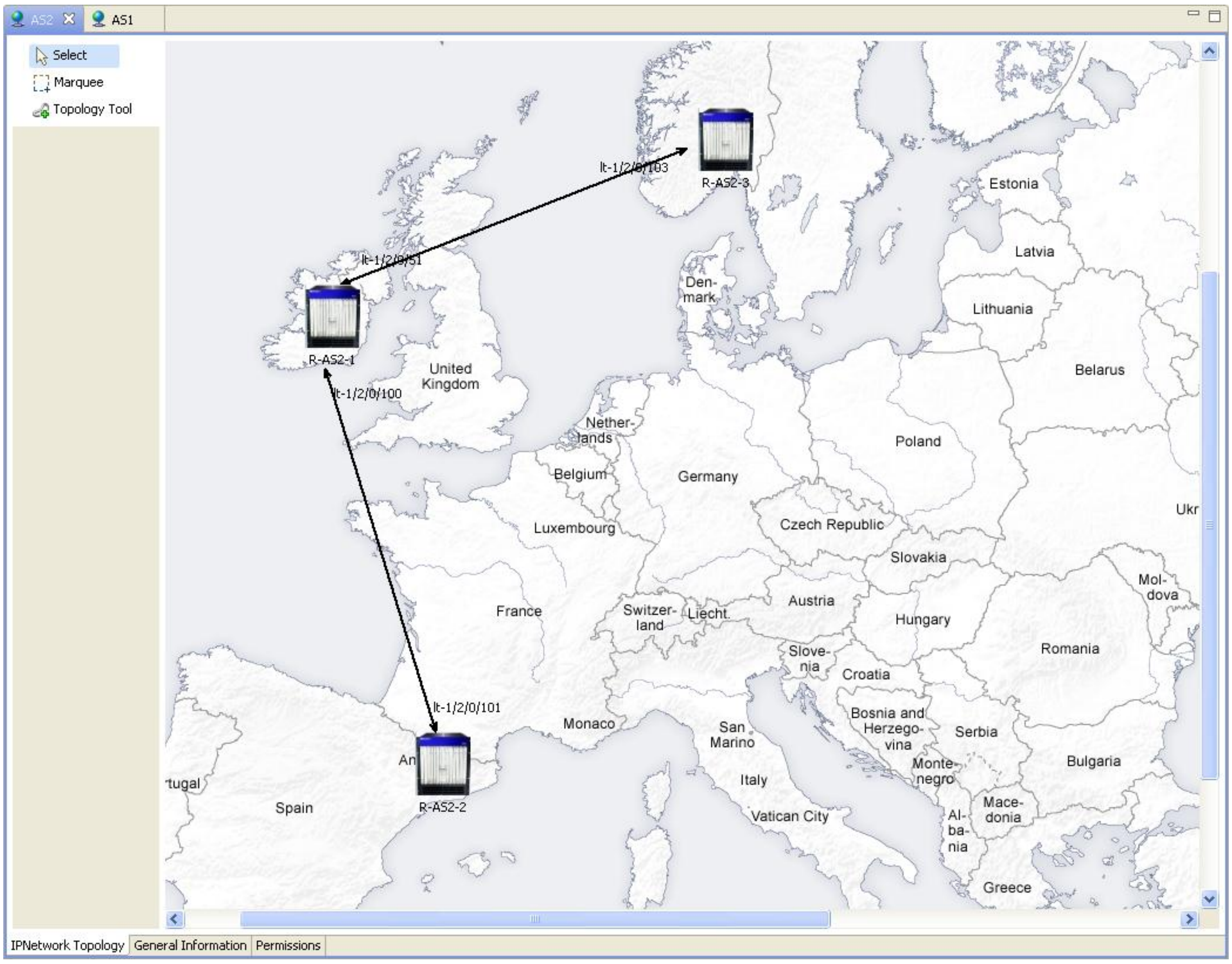


Figure 2.3: MANTYCHORE IP Network editor

2.3.5 Multi-domain support

The networks that MANTYCHORE sets up and manages are single-domain networks. A single operator manages the infrastructure and the resulting logical networks are managed by a single user. That said, the underlying infrastructure can in principle be provided by multiple administrative domains, in particular in the case of physical routers in physically diverse locations linked by trans-national links.

2.3.6 Testbed implementation and availability

MANTYCHORE does not plan to set up a publicly available testbed, although in the future a router may be available so that interested users can try the MANTYCHORE software in a demo situation. However, the project does plan to set up a testbed comprising a number of routers for the development, testing and demonstration of their own use case implementations.

2.3.7 Current status and roadmap

The MANTICORE II project has closed and its main outcomes have been ported to MANTYCHORE. Currently MANTYCHORE is upgrading its core framework to IaaS Framework and delivered an initial version in June 2011. For information about the project's future plans, please see [MANTYCHORE].

2.3.8 References

[Argia]	E. Grasa, S. Figuerola, A. Forns, G. Junyent, J. Mambretti, "Extending the Argia Software with a Dynamic Optical Multicast Service to support High Performance Digital Media", accepted for publication in Elsevier journal of Optical Switching and Networking Volume 6, Issue 2, April 2009
[MANTYCHORE]	MANTYCHORE website http://www.mantychore.eu/
[MANTYCHORE-DoW]	MANTYCHORE "Description of Work" http://jira.i2cat.net:8090/download/attachments/3211820/MANTYCHORE+FP7+-+DoW+-+Part+B+-+final+-+budget+removed.pdf

2.4 Phosphorus (UCLP)

The information about Phosphorus (User-Controlled Lightpath Provisioning (UCLP)) is unchanged. Please refer to [GN3-DJ1.4.1] Section 2.4.

2.5 4WARD

The information about 4WARD is unchanged. Please refer to [GN3-DJ1.4.1] Section 2.5.

2.6 GENI

Some of this summary, including the graphics, has been extracted from "GENI: Global Environment for Network Innovations – Facility Design" [GENI-GDD-07-44], dated March 2007, and was presented in [GN3-DJ1.4.1] (Section 2.6). However, new and/or updated information is provided in Sections 2.6.2.4, 2.6.4.5, 2.6.4.6, 2.6.4.7, 2.6.5 and 2.6.6.

2.6.1 Introduction

Global Environment for Network Innovation (GENI) [GENI] is a US programme funded by the NSF (National Science Foundation). It is an experimental facility designed to form a robust, federated environment to allow computer networks' researchers to experiment on a wide variety of problems in communications, networking, distributed systems, cyber-security, and networked services and applications with emphasis on new radical ideas. GENI will provide an environment for evaluating new architectures and protocols, over fibre-optic networks equipped with state-of-the-art optical switches, novel high-speed routers, radio networks, computational clusters and sensor grids.

GENI infrastructure presents some key characteristics in order to enable advanced research:

- Programmability: researchers can fully control GENI nodes' behaviour.
- Virtualisation: researchers can simultaneously share the GENI infrastructure using their own isolated slice of resources.
- Federation: different parts of the GENI infrastructure are owned and/or operated by different organisations.
- Slice-based experimentation: each experiment will be implemented on a specific slice of the GENI resources.

This experimental facility should pave the way to:

- Long-running, realistic experiments with enough instrumentation to provide real insights and data.
- Propose an infrastructure that promotes and makes adhesion easy for real users into these long-running experiments.
- Enable large-scale growth for successful experiments, so good ideas can be validated on a large scale.

Ultimately, GENI's goal is to avoid technology "lock in," enable addition of new technologies as they mature, and potentially grow quickly by incorporating existing infrastructure into the overall "GENI ecosystem".

A great number of projects are currently ongoing that are targeted at designing and operating prototypes of the GENI infrastructure. These projects are managed by the GENI Project Office (GPO).

2.6.2 Architecture overview

The high level GENI architecture can be divided into three levels, as shown in Figure 2.4:

- Physical substrate: represents the set of physical resources that constitute the GENI infrastructure, such as routers, links, switches.
- User services: represent the set of services that are available for the users in order to fulfill their research goals.

- GENI Management Core (GMC): defines a framework in order to bind user services with underlying physical substrate. In order to implement this, it includes a set of abstractions, interfaces and name spaces and provides an underlying messaging and remote operation invocation framework.

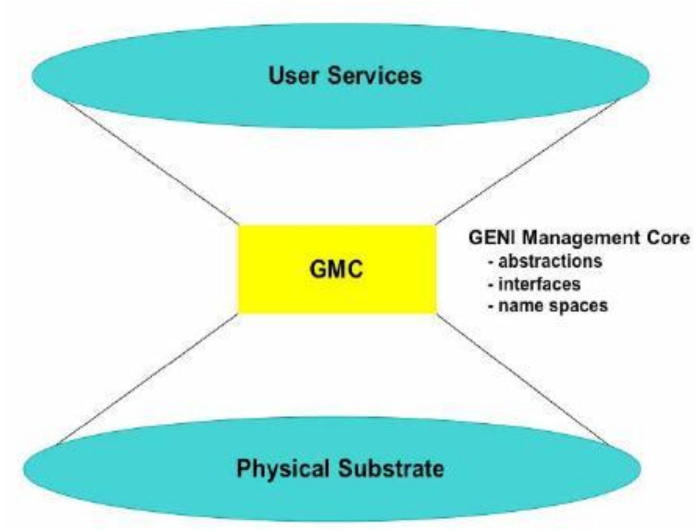


Figure 2.4: GENI architecture

The following sections provide more detail on the three levels of the GENI architecture.

2.6.2.1 *Physical substrate*

The physical substrate consists of an expandable collection of components. GENI components fall into one of the following categories:

- Programmable Edge Clusters (PEC): provide computational and storage resources as well as initial implementations of new network elements.
- Programmable Core Nodes (PCN): provide high-speed core-network data-processing functions.
- Programmable Edge Nodes (PEN): provide data-forwarding functionality at the boundary between access networks and a high-speed backbone.
- Programmable Wireless Nodes (PWN): implement proxies and other forwarding functionality within a wireless network.
- Client Devices: provide access to experimental services for end users.
- National Fibre Facility: provides 10 Gbps to 40 Gbps light path interconnection between PCNs.
- Tail circuits: interconnect GENI edge sites to the GENI core.
- Internet Exchange Points: interconnect the nationwide infrastructure to the commodity Internet.
- Urban 802.11-based Mesh Wireless Subnets: provide support for ad-hoc and mesh-network research.
- Wide-Area Suburban 3G/WiMax-based Wireless Subnets: provide open-access 3G/WiMax radios for wide-area coverage, along with short-range 802.11 class radios for hotspot and hybrid service models.

- Cognitive Radio Subnets: support experimental development and validation of emerging spectrum allocation, access, and negotiation models.
- Application-Specific Sensor Subnets: support research on both underlying protocols and specific applications of sensor networks.
- Emulation Grids: allow researchers to introduce and utilise controlled traffic and network conditions within an experimental framework.

2.6.2.2 *GMC*

The GENI Management Core (GMC) is a framework that defines a set of abstractions, interfaces and name spaces that binds together the GENI infrastructure. Because GENI's physical substrate and user services will develop and evolve rapidly as the facility is constructed and used, the GMC is designed to provide a narrowly defined set of mechanisms that both support and foster this development while isolating developmental change in one part of the system from that in other parts, so that independent progress may be made.

Abstractions

The GMC defines three key abstractions: components, slices, and aggregates. This sub-section introduces the abstractions; the following section describes the interfaces they support.

- **Components:** A component encapsulates a collection of substrate resources that can be either included on a single device or includes resources from many devices. Any resource can belong to only one component. Each component is controlled via a component manager (CM), over a well-defined interface. At the GENI facility it is possible to slice component resources among multiple users. This can be done either by virtualising component resources or by strictly partitioning them among the users. In both cases, the user is granted a sliver of the component. Each component is assigned a unique identifier as well as a human-friendly name.

- **Slices:** A slice is a set of slivers across a set of GENI components and an associated set of researchers that are implementing an experiment over these slivers. Each slice is assigned a unique identifier as well as a human-friendly name. Within the GENI framework, an experiment is a researcher-defined use of a slice.

- **Aggregates:** An aggregate is an unordered collection of components. Aggregates support hierarchy; an aggregate can contain other aggregates as well as components. Each aggregate has a unique identifier as well as a human-friendly name. Moreover, aggregates are controlled by aggregate managers.

Interfaces

GMC defines unique identifiers, called GENI Global Identifiers (GGID) for all the objects that constitute the GENI infrastructure, that is, components, slices and aggregates. A GGID is represented as an X.509 certificate.

Moreover, GMC defines two basic data types:

- **Resource specification (RSpec):** the data structure that describes GENI resources. It contains information about the resources that are encapsulated by components, their processing capabilities, their network interfaces and the instrumentation available on them.
- **Tickets:** granted by a component owner to a researcher, and later "redeemed" to acquire resources on the component.

GMC defines a series of operations for components, slices and aggregates. Some of them are mentioned below:

- Creating/modifying/deleting slices.
- Request for allocating a slice.
- Start/stop/delete a slice.
- Add/delete components in an aggregate.

2.6.2.3 *User services*

As shown in Figure 2.4, user services are built on top of the GMC and are the set of distributed services that enable GENI users to implement experiments on a given slice as well as to manage their allocated slices.

From the user services point of view, diverse user communities are defined in GENI. These communities are:

- Owners of parts of the substrate: define usage policies of the substrate and provide mechanisms for enforcing these policies.
- Administrators of parts of GENI: manage the GENI substrate ensuring proper operation.
- Developers of user services: create GENI services using the GMC interfaces.
- Researchers: use the GENI facility in order to conduct research. They can allocate resources on the GENI substrate and deploy specific software.
- End users not affiliated with GENI, but who access services provided by research projects that run over GENI.
- Third parties that may be impacted from GENI operation.

2.6.2.4 *GENI system overview*

A block diagram of the overall GENI system covering the most important entities is shown in Figure 2.5 below.

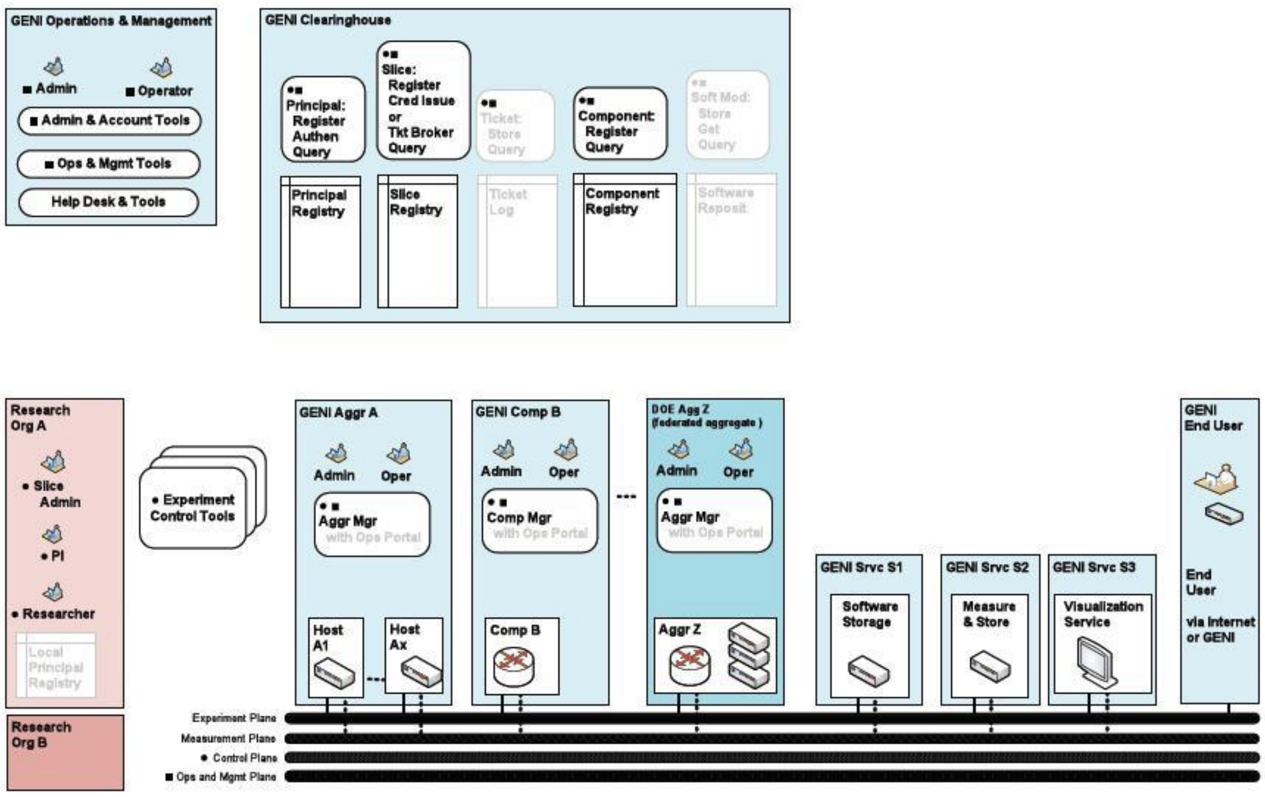


Figure 2.5: GENI block diagram or GENI-System [GENI-Intro]

The clearinghouse is a centralised software entity that registers all the GENI elements. It is a cornerstone of the GENI system, since each GENI user needs to communicate with it in order to request GENI resources or modify the set of resources that are available to him/her. More specifically it includes:

- Principal registry, which holds a record for each GENI actor (e.g. researcher, administrator).
- Slice registry, which holds a record for each slice including information regarding the responsible organisation, and slice status.
- Component registry, which holds a record for each affiliated substrate component or aggregate that is part of the GENI system.

2.6.3 User community

As stated in Section 2.6.1 *Introduction* on page 15, the GENI infrastructure will provide the opportunity for researchers to experiment on a wide variety of innovative ideas on computer networks.

2.6.4 Mechanisms for providing virtualisation

2.6.4.1 *Implementation of virtualisation on Layer 3*

Each Programmable Core Node (PCN) includes a Packet Processing System (PPS) which is a collection of line cards, general-purpose processors, and programmable hardware (e.g., network processors and Field Programmable Gate Arrays (FPGAs)) connected via a switch fabric. PPS implements a high-speed programmable device that supports multiple virtual routers, possibly belonging to different slices, within a shared platform. The term virtual router is used to denote any network element with multiple interfaces that forwards information through a network, while possibly processing it as it passes through. As such it also encapsulates the functionality of a conventional Ethernet switch. PPS design has two main goals:

- To provide the necessary resources to the researchers in order to build their own virtual routers that can operate at high speed.
- To ensure that virtual routers operating in different slices will run without interference.

The design of the PPS is quite different from the design of conventional routers and switches in that it must allow bandwidth to be flexibly allocated among multiple virtual routers and provide third-party access to generic processing resources that can be flexibly allocated to different virtual routers. Hence, PPS design requirements include open hardware and software components, scalable performance, stability and reliability, ease of use, technology diversity and adaptability, flexible allocation of link bandwidth and strong isolation between virtual routers.

2.6.4.2 *Implementation of virtualisation on Layer 2*

Each Programmable Core Node (PCN) also includes a Circuit Processing System (CPS), which is a layered collection of circuit-oriented elements, as shown in Figure 2.6.

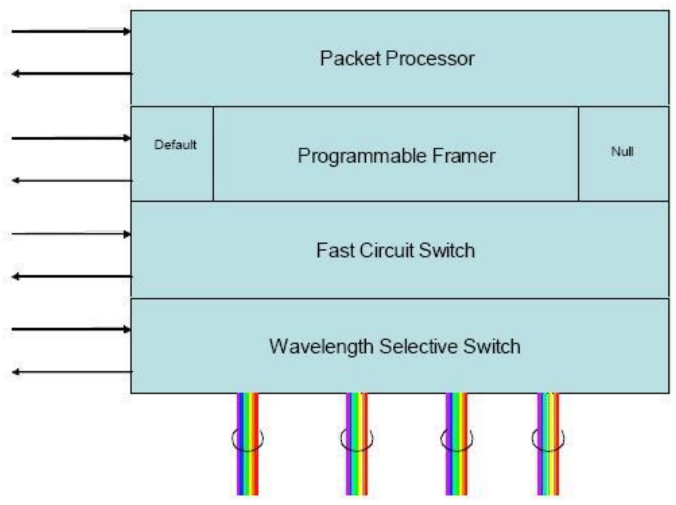


Figure 2.6: CPS design

Researchers can access CPS at whatever layer provides the appropriate of abstraction for their work. The layers of the CPS are described below:

- Wavelength Selective Switch (WSS): Data on one 10 Gbps wavelength can be switched to another wavelength, or delivered to the Fast Circuit Switch. Data carried on a wavelength is totally transparent for the WSS. User research equipment can connect directly to the WSS (shown on the left hand side of Figure 2.6).
- Fast Circuit Switch (FCS): Circuits are multiplexed using TDM onto a 10 Gbps wavelength. Virtual circuits of any bandwidth with granularity 1 Mbps can be established. Individual circuits can be assembled from multiple basic slots within and across wavelengths. The FCS will connect to the WSS via optical fibre. User research equipment can connect directly to the FCS (shown on the left hand side of Figure 2.6).
- Programmable Framer (PF): The framer will frame packets inside circuits. SONET is used for default framing format. The framer should have a null framing format in cases where the packets themselves carry sufficient information for recovery at the destination. The PF will connect to the FCS via electrical links or short-reach optics. User research equipment can connect directly to the PF (shown on the left hand side of Figure 2.6).
- Packet Processor (PP): this corresponds to the PPS subsystem described above. PP can bypass the PF layer and be connected directly to the FCS layer using optical fibres.

The CPS design is planned to be implemented using commercially available hardware.

2.6.4.3 *Implementation of virtualisation on Layer 1*

Layer 1 virtualisation issues are covered in previous section (2.6.4.2)

2.6.4.4 *Implementation of computing virtualisation*

Computing and storage services are provided by Programmable Edge Clusters (PECs). GENI plans to deploy PEC components at 200 different sites on the GENI infrastructure.

PECs will consist of a rack equipped with commodity processors, high storage capacity, and connection to the local network infrastructure. Each PEC will run virtualisation software that will implement slivers as virtual machines (VM), each of which can be bound to some amount of processor, memory, disk, and link capacity under the control of the Component Manager (CM). Two different virtualisation technologies are expected to be deployed:

- Paravirtualisation, which gives slivers access to low-level hardware resources.
- Container-based virtualisation, which gives slivers access to a virtualised system call interface.

While PECs emulate computational clusters, they may also act as clients, individual servers, server farms, ingress routers for testing of innovative network architectures, etc. Storage capacity and computational capability of PECs will differ significantly from site to site. At the low end, a PEC will include 8-12 processors and at the high end, a PEC might include 512-1024 processors.

2.6.4.5 *OpenFlow-based virtualisation*

GENI has recently adopted the OpenFlow technology for slicing (virtualising) its experimental infrastructure. OpenFlow is an attractive technology for GENI users as it allows decoupling of the forwarding and decision-making parts of a network element while giving full control over the decision-making (control) part to the user/experimenter [OpenFlow2]. In a network infrastructure utilising OpenFlow technology, virtualisation is enabled by deploying FlowVisor [FlowVisor], which allows multiple logical networks, each with different addressing and forwarding schemes, to co-exist on the same physical infrastructure. A FlowVisor is a decision-making entity within a network that maintains a policy engine, translation and forwarding mechanisms to fulfil the following virtualisation goals:

- Isolation: Multiple virtual segments are created and allocated to user controllers within the same physical substrate. Each virtual network segment is independent and hence gives users secure isolation.
- Scalability and flexibility: FlowVisor supports the creation of highly diverse virtual networks, keeping in mind the allocated resources like bandwidth, traffic, CPU, etc.

An experimental infrastructure utilising OpenFlow technology can be integrated within the GENI control framework by adopting a specific aggregated manager, which is FlowVisor OpenFlow Aggregate Manager (FOAM).

2.6.4.6 *Management of virtualised infrastructure*

The GENI infrastructure will be managed by the GENI operator by means of an operator portal. According to the ITU Fault, Configuration, Accounting, Performance, Security (FCAPS) model, the following management functionality is planned to be offered to the GENI operators:

- Fault management. GENI operators will be able to detect and repair problems on the GENI infrastructure.
- Configuration management. GENI operators will be able to provision, configure and validate new components of the GENI infrastructure.
- Accounting management. GENI operators will ensure that only authorised users and experiments can use the GENI infrastructure as well as deploy policies on the usage of the infrastructure.
- Performance management. GENI operators will be able to monitor the utilisation and performance of the GENI components.
- Security management. GENI operators will receive security-related information on the usage of the GENI infrastructure and will be able to find out whether GENI is being attacked or the GENI Acceptable Use Policy is being violated.

According to [GENI-SFA], each component or aggregate supports a management interface that is used to boot and configure them. This interface supports at least the following operations:

- SetBootState(Credential, State)

  Used to set the boot state of a component to one of the following four values: debug (component trying to boot), failure (hardware failure), safe (component available only for operator diagnostics), and production (component available for hosting slices).
- State = GetBootState(Credential)

  Used to learn a component's boot state.
- Reboot(Credential)

  Forces the component to reboot.

2.6.4.7 *Control of virtualised infrastructure*

Each slice has an interface that is used for creation and control. More specifically, each authorised user is able to add or remove resources to the slice by using this interface.

2.6.4.8 *Implementation of user interface*

Researchers will interact with the GENI infrastructure via the researcher portal which allows researchers and developers to specify the characteristics of their experiments and manage the experiments themselves. More specifically the researcher portal is going to be the front-end of a set of services offering the following functionality:

- Resource allocation: defines how the resources are shared among experiments (acquired, scheduled, or released).
- Slice embedding: instantiates a researcher's slice over a number of components.
- Experimenter workbench: provides a set of tools to create, configure and control researchers' experiments.

2.6.5 Multi-domain support

By design, GENI can federate with other virtualisation frameworks, either following the GENI approach or not. According to the GENI design, the interconnection will be implemented via the GENI clearinghouse entity and provides a narrow set of interfaces that the other virtualisation framework must follow. Moreover, a basic GENI assumption is that there will be multiple owners of the physical substrate in a federated fashion, forming the entirety of the infrastructure.

2.6.6 Testbed implementation and availability

Currently, GENI's experimental test facilities comprise the following main parts:

- Backbone networks. These include major research networks in the USA interconnecting experimental GENI testbeds. The backbone networks are:
  - Internet2. Internet2 provides the US research and education community with a dynamic hybrid optical and packet network. GENI experimenters have access to 1 Gbit/s of dedicated bandwidth from Internet2. Experimenters may create their own topologies using Layer 2 VLANs.
  - National Lambda Rail (NLR). NLR provides the testbed for advanced research at over 280 universities and private and US government laboratories. GENI experimenters have access to up to 30 Gbit/s of non-dedicated bandwidth on NRL. Experimenters may create their own topologies using Layer 2 VLANs.
  - GENI OpenFlow Core. The GENI network core is a set of OpenFlow-capable switches in NLR and Internet2. There are currently two standing VLANs (3715 and 3716) carried on ten switches in the core. Experimenters may use these standing VLANs within the GENI core network without having to coordinate with NLR or Internet2. Experimenters will however have to coordinate with their campus and/or regional networks to connect to the GENI core. The two standing VLANS in the network core also bridge between the Internet2 and NLR networks.
- Programmable hosts. These include a set of computing platforms and clients available for GENI experimenters.
- Programmable networks: These are a set of local wired experimental networks capable of hosting experimental tests. They are mainly based on either OpenFlow or PlanetLab technologies.
- Wireless testbeds. Currently a limited number of local wireless testbeds (mainly based on Wi-Fi technology) are available for GENI experimenters.

2.6.7 Road map

Currently GENI is working on finalising the specification for an experimental facility to join the GENI infrastructure. At the same time, GENI developers are working on implementing new aggregate managers to support more diverse technologies such as WiMax.

2.6.8 References

[GENI]	http://www.geni.net
[GENI-GDD-07-44]	L. Peterson (ed.), "GENI: Global Environment for Network Innovations – Facility Design", GDD-07-44, March 2007
[GENI-Overview]	Larry Peterson, Robert Ricci, Aaron Falk, Jeff Chase, "The Global Environment for Network Innovations (GENI)", April 2009 http://www.geni.net/wp-content/uploads/2009/04/geni-at-a-glance-final.pdf

[GENI-SFA]	Larry Peterson, Robert Ricci, Aaron Falk, Jeff Chase “Slice-Based Federation Architecture”, 2010, GENI WIKI
[GENI-Intro]	Harry Mussman, “GENI: An Introduction”, 2011, GENI WIKI
[GENI-System]	[details to be provided]
[GN3-DJ1.4.1]	M. Campanella, P. Kaufman, F. Loui, R. Nejabati, C. Tziouvaras, D. Wilson, S. Tyley, “Deliverable DJ1.4.1: Virtualisation Services and Framework Study” http://www.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-09-225%20DJ1.4.1v1.0%20Virtualisation%20Services%20and%20Framework%20Study.pdf
[OpenFlow2]	Nick McKeown, et al., “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication, Apr 2008
[FlowVisor]	Rob Sherwood, et al., “FlowVisor: A Network Virtualization Layer”, Oct 2009 www.openflow.org

2.7 PlanetLab/VINI/OneLab

The information about PlanetLab/VINI/OneLab is unchanged. Please refer to [GN3-DJ1.4.1] Section 2.7.

2.8 AKARI

Note that at the time of writing this document, only limited information is publicly available for the AKARI project, in particular, a conceptual design document [AKARI-ConceptualDesign]. For this reason, no concrete information on the project’s achievements can be provided, only information regarding AKARI’s vision.

Additional information about the virtualisation aspects that AKARI will support are available from a paper written by Akihiro Nakao (University of Tokyo) [Nakao1] and from a talk by Kiyohide Nakauchi (NICT) [Nakauchi] about the implementation choice that will likely be adopted once AKARI reaches its implementation phase. It is therefore important to remember that the information about the virtualisation strategy reported here is not an official statement from AKARI but a summary of the indications provided by AKARI’s partners.

2.8.1 Introduction

The Internet is the object of various projects whose aims are to identify the limits of the current networking models and propose alternatives to circumvent them. There are two approaches when dealing with Future Internet (FI) development. The first one is to develop enhancements with reference to the current situation. Therefore some limitations have their counterpart answers. For example, IPv6 was initially developed in order to cope with IPv4 address depletion, while mobile IP is meant to extend users’ mobility. An alternative is the clean slate approach, where the IP protocol, whether IPv4 or IPv6, is not even part of the answer. AKARI is the Japanese initiative related to Future Internet networks, the GENI and 4WARD projects being the US and EU counterparts respectively.

AKARI’s vision considers virtualisation and particularly network virtualisation as a major technology enabling the possibility to deploy various large-scale network ecosystems in parallel. As network virtualisation pushes the limit of node virtualisation up to the core infrastructure, it is now even possible to deploy several instances of potential Internet networks without relying on the Internet itself.

Network virtualisation would therefore promote:

- Competition between these potential network instances.
- Cooperation between these network instances.
- Easier comparison between these instances as they exist in parallel.
- Relevant experiment results as the technology can enable several large-scale testbed deployments worldwide without incurring additional infrastructure cost.

2.8.2 Architecture overview

[AKARI-ConceptualDesign] describes high-level criteria for designing next-generation networks, using virtualisation technologies among others.

Section 4.12 in [AKARI-ConceptualDesign] states that “Research on network services and architectures using overlay networks has been popular recently in various countries. In particular, overlay network testbed infrastructure is being installed at PlanetLab in the US, OneLab/OneLab2 in Europe, as well as in Japan (the University of Tokyo and NICT), Germany, China, and Korea.”

The reasons behind the adoption of virtualisation technologies include the large-scale and geographically dispersed nature of their networks, their inherent reliability in case of failure (thanks to easy cloning of virtual resources), and the reduction of maintenance costs. They “enable the threshold (cost) for visiting a new network business to be reduced by sharing this demonstration experimental environment. Therefore, an overlay network testbed has great social significance as a core technology for creating innovative services and performing early-stage development” [AKARI-ConceptualDesign].

[AKARI-ConceptualDesign] goes on to consider the different evolution patterns for a testbed based on virtualised networks. However, no indications on how the virtualisation will be implemented are provided.

Among the open issues listed in the document ([AKARI-ConceptualDesign] Section 4.12.4), those related to virtualisation include the following:

- Virtualisation layer. The layer in which virtualisation is to be performed has to be determined. Currently, a tunnelling technique is generally considered as the virtualisation method. However, the means of efficiently implementing tunnelling also has to be determined.
- Distributed administration of virtual domains (e.g. resource management and operational cost) has to be considered differently from the hierarchical approaches used for physical infrastructures.
- Node virtualisation and network virtualisation: “Node virtualisation, which virtualises new-generation routers or nodes that are co-located on routers, includes the recent operating system virtualisation, I/O

virtualisation (optimisation), and network virtualisation. Virtualisation of the network part must take the problem of how to isolate and allocate existing physical resources into consideration more than virtualisation." [AKARI-ConceptualDesign]

- Engineering: "Reusing node construction technology, which was fostered in the field of active networks, and linking it with operating system research will significantly contribute to the development of network virtualisation technology." [AKARI-ConceptualDesign]

More details on the adoption of virtualisation for a testbed network containing virtual routers are given in [AKARI-ConceptualDesign] Section 6.2. The section discusses the design of a virtual router, running on a Virtual Machine (VM) in an experiment. Node resources are abstracted in the data link layer (L2) and below (the document gives no indication of how), with the purpose of implementing new protocols or new architectures at the higher layers. The use case assumes that the virtual resources run in a completely isolated environment without interference or efficiency degradation due to the activities of concurrent experiments overlaying the same physical resources.

Additional requirements and open issues regarding the introduction of virtual routers in a testbed are introduced:

- Abstraction layer flexibility. "The layer where abstraction is performed (virtualisation is provided) must be able to be freely set for each VM. For example, there must be a means of enabling a new L2 technology experiment and a new L3 technology experiment to be executed at the same time (on separate VMs) within one physical router."
- Mapping to a lower-layer multiplexing technology. "A mapping policy must be determined between L2/L1 multiplexing technologies and resources that are isolated by network virtualisation at higher layers. […] At the same time, the VMs must have enough versatility (interface abstraction) to enable various lower layer technologies to be supported."
- Resource management. Specific information modelling, resource monitoring and scheduling technologies are required in order to map virtual resources on the physical substrate in an optimal way.
- Interconnection. The possibility to federate different virtual testbeds is required.

Some hints on what functionalities will be provided in AKARI network virtualisation can be deduced by the following statement: "network virtualisation [is assumed] as a technique for isolating computational and network resources […] to allocate them to a logical (virtual) network for accommodating multiple independent and programmable virtual networks. We can always add isolation of the other kinds of resources such as storage, but for the sake of simplicity, we intentionally do not include them" [Nakao1].

The document goes on to state the difference between VPN and network virtualisation by specifying the features that the latter should provide: "(1) programmability: a virtual network may be equipped with programmable control plane, (2) topology awareness: a virtual network may be topology-aware rather than offering only connectivity, (3) quick re-configurability: a virtual network may be quickly provisioned and reconfigured, (4) resource isolation: a virtual network may be allocated a set of computational and network resources, and (5) network abstraction: a virtual network may accommodate a new architecture different from the current Internet architecture." [Nakao1].

The concept of a slice that is adopted is very similar to the one available in FEDERICA, with a strong emphasis on isolation and reproducibility of the behaviour of the virtual resources: "A slice is an isolated set of computational and network resources allocated and deployed across the entire network. Slice consists of primitives such as a link sliver that conveys traffic, a node sliver that forks traffic with equipped programmability, and an interface that connects a link sliver and a node sliver. No matter what format of data is transmitted and conveyed on a slice, a node sliver can be programmed to parse, route and forward the data through a link sliver to another node sliver. In other words, the abstraction model allows us to define an arbitrary data format, whether to transmit data, e.g., via circuits or packets, how to route data, etc. within a slice. A final note in this section is that although the term includes virtualisation, our primary goal is to 'isolate' resources for an individual logical network using virtualisation techniques as a means, thus, virtualisation itself is not necessarily our goal." [Nakao1]

More hints about the implementation choices that AKARI could follow are reported in [Nakauchi]. Two possible approaches are described: one relying on software-only network virtualisation, and one aiming at the development of hardware devices capable of network virtualisation. The details about the two approaches and the testbeds currently available for these technologies are described in detail in [Nakao2].

2.8.3 User community

Currently there are no hints about the user community that could benefit from access to the testbeds.

2.8.4 Mechanisms for providing virtualisation

Two approaches are described for providing virtualisation [Nakao2]:

- CoreLab: a software-based network virtualisation approach that adopts commercial off-the-shelf x86 servers and open source virtualisation software.

  CoreLab supports different virtualisation methodologies: host-based hypervisors (KVM) and resource containers (OpenVZ, LXC). It also allows Network Interface Controller (NIC) exclusive access to host VMs via Peripheral Component Interconnect (PCI) pass through.
- VNode: a custom hardware device that enables network virtualisation through slicing of its cards and servers. A VNode is composed of a Programmer part and a Redirector module. The Programmer is made of high-end servers equipped with Fast-Path network processor cards and OpenFlow switches. The Redirector is a 10 Gbit/s production router, with additional service module cards, and routes packets according to the directives sent by the network processor cards.

2.8.4.1 *Implementation of virtualisation on Layer 3*

CoreLab implements virtualisation between slivers by using Generic Routing Encapsulation (GRE)-tap tunnels. The switching facilities are emulated via OpenFlow Switch In A Slice (OFIAS) and virtual OpenFlow Switches (vOFS) as slivers. The difference between the solutions is not stated.

The current VNode model supports GRE encapsulation. Developers state that it can be replaced with different L2/L3 VPN implementations (e.g. MPLS, VLAN, and OpticalPath).

2.8.4.2 *Implementation of virtualisation on Layer 2*

CoreLab supports VLAN tagging. However no details are provided on how it is applied for separating traffic.

The current VNode model supports GRE encapsulation. Developers state that it can be replaced with different L2/L3 VPN implementations (e.g. MPLS, VLAN, and OpticalPath).

2.8.4.3 *Implementation of virtualisation on Layer 1*

No information about this feature is available.

2.8.4.4 *Implementation of computing virtualisation*

The CoreLab approach is built on systems virtualisation, supporting KVM host-based hypervisor. Therefore, CoreLab can virtualise computing resources.

VNode's architecture includes four high-end servers capable of virtualisation but not intended for general purpose hosts. It is used to control the Redirector device, Slow path, and Fast path network processor cards.

2.8.4.5 *Management of virtualised infrastructure*

No details are available. See *Implementation of user interface* below.

2.8.4.6 *Control of virtualised infrastructure*

No details are available. See *Implementation of user interface* below.

2.8.4.7 *Implementation of user interface*

The CoreLab virtualisation approach is equipped with a GUI for editing slices' topology and a web-based VNC for logging in a slice.

The VNode devices are equipped with a software control pane, while configurations for slivers, links, and VNode's interfaces are provided as an XML configuration file.

2.8.5 **Multi-domain support**

The available conceptual design document [AKARI-ConceptualDesign does not indicate whether a multi-domain scenario will be considered in AKARI. One of the Points of Presence (PoPs) of the virtualisation testbeds is located in South Korea, but no information on how the two domains are managed is reported.

2.8.6 Testbed implementation and availability

There is one testbed for CoreLab network virtualisation: it shares the substrate of PlanetLab Japan with additional network virtualisation support, involves more than 24 nodes, and is overlaid on multiple backbone networks. No information about the availability of the testbed is given.

Four VNodes were deployed in September 2010. They are connected through JGN2Plus and JGN-X.

2.8.7 Current status and roadmap

Currently there are no indications about the roadmap for integrating the testbeds described in [Nakao2] into the AKARI production environment.

2.8.8 References

[AKARI-ConceptualDesign]	"New Generation Network Architecture: AKARI Conceptual Design" http://akari-project.nict.go.jp/eng/concept-design/AKARI_fulltext_e_preliminary_ver2.pdf
[Nakao1]	Akihiro Nakao, "Network Virtualization as Foundation for Enabling New Network Architectures and Applications", IEICE Transactions on Communications, Volume E93.B, Issue 3, pp. 454-457 (2010), available at adsabs.harvard.edu/abs/2010IEITC..93..454N
[Nakao2]	Akihiro Nakao, "Architectures and tTestbeds eEnabled tThrough aAdvanced nNetwork vVirtualization: CoreLab and VNode pProjects", 3rd EU- Japan Symposium on Future Internet http://ec.europa.eu/information_society/activities/foi/research/eu-japan/eujapan3/docs/nakao.pdf
[Nakauchi]	Kiyohide Nakauchi, "Introduction to Network Virtualization Technologies in Future Internet Research", Asia FI Summer School (August 26, 2010), available at www.asiafi.net/meeting/2010/summerschool/p/nakauchi.pdf

2.9 GEYSERS

The information in this section is based on three GEYSERS deliverables: "D2.1 Initial GEYSERS Architecture and Interfaces Specification", "D3.1 Functional Description of the Logical Infrastructure Composition Layer (LICL)" and "D4.1 GMPLS+/PCE+ Control Plane Architecture" [GEYSERS-D2.1, GEYSERS-D3.1, GEYSERS-D4.1].

2.9.1 Introduction

GEYSERS' vision is to qualify optical infrastructure providers and network operators with a new architecture, to enhance their traditional business operations. Optical network infrastructure providers will compose logical infrastructures and rent them out to network operators; network operators will run cost-efficient, dynamic and mission-specific networks by means of integrated control and management techniques. In the GEYSERS concept, high-end IT resources at users' premises are fully integrated with the network services procedures, both at the infrastructure planning and connection provision phases.

GEYSERS will define and implement a novel photonic network architecture, capable of provisioning “optical network + any-IT” resources to network operators for end-to-end service delivery. GEYSERS proposes a revolutionary vision under an evolutionary approach that follows a network-centric and bottom-up strategy. This vision is based on partitioning the photonic network infrastructure to create specific logical infrastructures. Each logical infrastructure will be controlled by an enhanced Network Control Plane capable of provisioning Optical Network Services bundled with IT resources on an on-demand basis. Furthermore, the logical composition of photonic networks will enable the GMPLS/ Path Computation Element (PCE) control plane to dynamically scale infrastructure resources based on the Network Operator's needs.

2.9.2 Architecture overview

The GEYSERS architecture presents an innovative structure by adopting the concepts of Infrastructure as a Service (IaaS) and service-oriented networking to enable infrastructure operators to offer new IT + network converged services. In the GEYSERS architecture, the service-oriented paradigm and IaaS framework enable flexibility of infrastructure provisioning in terms of configuration, accessibility and availability for the user. At the same time, the layer-based structure of the architecture enables the separation of the functional aspects of each of the entities involved in the converged service provisioning, from the service consumer to the physical ICT infrastructure. Figure 2.7 shows the GEYSERS architecture reference model; each layer is responsible for implementing different functionalities covering the full end-to-end service delivery, from the service layer to the physical substrate.

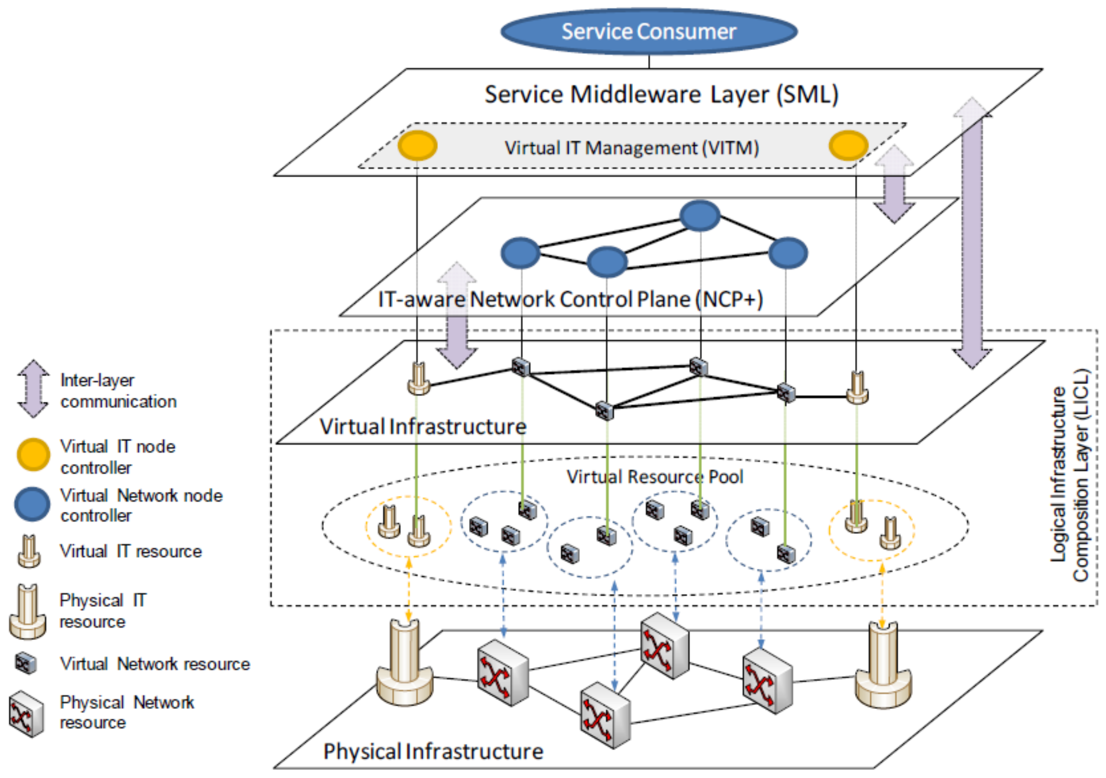

Service Consumer
Service Middleware Layer (SML)
Virtual IT Management (VITM)
IT-aware Network Control Plane (NCP+)
Virtual Infrastructure
Virtual Resource Pool
Logical Infrastructure Composition Layer (LICL)
Physical Infrastructure
Inter-layer communication
Virtual IT node controller
Virtual Network node controller
Virtual IT resource
Physical IT resource
Virtual Network resource
Physical Network resource

Figure 2.7: GEYSERS architecture

The Network Control Plane (NCP) is proposed as an extension of ASON/GMPLS and PCE, both in terms of architectural elements and protocol objects/procedures. The NCP layer is responsible for the dynamic provisioning of network connectivity services in support of the IT services managed by the Service Middleware Layer (SML). The NCP is also in charge of the control/management of the logical network infrastructure composed by the Logical Infrastructure Composition Layer (LICL), seen and controlled just as a physical infrastructure. Since each logical infrastructure, controlled by an NCP instance, can include both network and IT resources, the NCP strictly cooperates with the SML in order to coordinate and optimise the combined utilisation of network and IT resources. In particular, the NCP is in charge of dynamic network service provisioning, monitoring and recovery functions.

The novel Logical Infrastructure Composition Layer (LICL) allows/supports the partitioning of the physical infrastructure, including both optical network and IT resources. It utilises a semantic resource description and information modelling mechanisms for concealing the technological details of the physical layer from network operators. Logical resources are represented seamlessly using a standard set of attributes which allows the Control Plane to overcome the network and technology segmentation. Partitioning provides a 1:N logical representation of a physical resource from one or multiple domains. The Logical Infrastructure Composition Layer allows dynamic and consistent monitoring of the physical layer and binding/associating the right security and access control policies. Furthermore, this layer constitutes application-specific logical infrastructures by interconnecting the logical resources based on the virtual infrastructure operators' requirements.

The Service Middleware Layer (SML) acts as an intermediate layer between applications running at the service consumer's premises and the enhanced NCP; it is able to translate the application requests and Service Level

Agreements (SLAs) into IT service descriptions specifying the associated network and IT resources and triggers the provisioning procedures at the NCP. All service requests from the application side will be handled by the SML and translated to technology-specific requests before the provisioning of services over the Virtual Infrastructure.

At the lowest level in the GEYSERS architecture there is the Physical Infrastructure layer that comprises optical network and IT resources from different Physical Infrastructure Providers.

The GEYSERS architecture shows how physical devices are partitioned and abstracted into virtual resources that can be grouped logically, without order, as a Virtual Resource Pool (VRP). Virtual resources in the VRP can then be selected and composed, creating a Virtual Infrastructure using the tools that the LICL provides to Physical and Virtual Infrastructure Providers (PIP, VIP). Controllers in the NCP configure and manage the virtual network resources; similarly, Virtual IT Node controllers at the Virtual IT Manager (VITM) configure and manage virtual IT resources. The SML lies on top, offering the converged services.

2.9.3 User community

There is no user community behind GEYSERS. GEYSERS' goal is to exploit infrastructure providers' physical resources.

2.9.4 Mechanisms for providing virtualisation

Work in this area is in progress. Final results are not yet available, though early results are expected to be ready in the near future. However, the focus is on virtualisation mechanisms for the optical layer only.

2.9.4.1 *Implementation of virtualisation on Layer 3*

The research carried out in GEYSERS does not include a study of virtualisation on Layer 3.

2.9.4.2 *Implementation of virtualisation on Layer 2*

The research carried out in GEYSERS does not include a study of virtualisation on Layer 2.

2.9.4.3 *Implementation of virtualisation on Layer 1*

Optical network virtualisation is the creation of logical instances of optical network resources whose behaviour is the same as their corresponding physical optical network resources. It enables multiple Optical Virtual Network Infrastructures (Op-VNIs) over the same physical substrate while isolating them from each other. It is achieved by partitioning a single physical resource or aggregating multiple physical resources. Optical network virtualisation can support various granularities, including sub-wavelength, wavelength and waveband, as well as any combination of these. In each virtual optical network infrastructure, different granularities can be supported, such as sub-wavelength, wavelength, waveband, or a mix of different granularities. Optical network virtualisation in GEYSERS relies on the abstraction of heterogeneous network resources, including nodes, links and segments comprising both nodes and links.

Optical node virtualisation is a procedure that represents the optical nodes as virtual instances that inherit critical characteristics from the physical entities. It relies on either the partitioning of a single optical node or the aggregation of multiple optical nodes. Each virtual optical node has its own ports and switch capability.

Similar to optical node virtualisation, optical link virtualisation abstracts optical fibre links as virtual instances by partitioning and/or aggregation. The partitioning of optical fibre links results in the granularities of sub-wavelength and wavelength while the aggregation results in a granularity of waveband.

Several physically disjoint optical links or portions (slices) of links can be aggregated and virtualised together. In this case the intermediate associated optical nodes or portions (slices) of node, which are required to interconnect the physically disjoint links, will also need to be included in the aggregation of resources to be virtualised. The resultant network virtual resource will be, in this case, a single virtual link in the virtual network infrastructure.

2.9.4.4 *Implementation of computing virtualisation*

The virtualisation of IT resources in the GEYSERS project is still in the research phase. IT resources, in the context of GEYSERS, are computing and storage nodes running user applications that are interconnected by a virtualised network infrastructure. Users can request such IT resources that are in reality abstractions or partitions of real physical resources. GEYSERS does not propose any innovative approach to IT virtualisation, but exploits existing mechanisms for implementing virtual IT resources using the following paradigms:

- Abstraction (1:1). A physical IT resource can be exposed as a whole to the user who can customise it and deploy his software. A user can, for example, reserve pre-installed physical servers and use them to install and run any applications, as is the case in experimental facilities and grid environments. Storage-only nodes can, for example, be exposed as 1:1 abstractions from a Network Attached Storage (NAS).

- Partitioning (1:N). An IT resource can be partitioned into N virtual IT resources using common virtualisation technologies, such as Xen, KVM, VMware, VServer, etc., where each partition is represented as a virtual machine (VM) with computing and storage resources. These technologies use different types of virtualisation to partition the resources. While OS-level virtualisation (e.g. VServer) offers interesting performance, it allows only limited isolation and customisation. By contrast, performing emulation and hardware virtualisation (e.g. KVM, Xen), each VM has its own isolated execution environment where any OS can run. Especially using the hardware virtualisation features of current processors (Intel-VT, Amd-V), near to native computing performance can be obtained inside virtual machines. As a drawback, device access, such as disk and network, is more costly as the device needs to be emulated and accessing it involves the translation of all the instructions. This is where para-virtualisation (e.g. Xen, KVM/Virtio) helps, exposing virtual device drivers through which virtual machines can access the hardware devices with only minimal overhead. Nevertheless, with para-virtualisation the OS of the VM needs to be modified in order to support the specific virtual drivers. For storage, technologies such as Storage Area Network (SAN) and Network Attached Storage (NAS) can be used to expose storage to be shared into virtual disks and used as different virtual IT resources. Here, virtualisation is performed through the network, for example using Internet Small Computer System Interface (iSCSI) protocol. Another solution is the Shared Storage Model (SSM) of the Storage Network Industry Association (SNIA), which allows a storage device to be divided into several individual ones, each using a different storage technology, and isolating the different virtual storages. Using Network File System (NFS) or Common Internet File System (CIFS), several machines can access the same physical storage device through the network, hence sharing a common file system.

- Aggregation (N:1). In contrast to partitioning, aggregation consists of exposing a set of physical IT resources as a single virtual IT resource to the user. Such aggregation is possible with Versatile SMP (vSMP), for example, which aggregates many physical servers and makes them appear to the OS like one giant machine with many cores. Regarding storage nodes, it is possible to aggregate different disks into a common logical storage pool. This can also be done using SNIA technology, allowing not only a device to be divided into several individual ones, but also the aggregation of several physical devices to make them appear as one single virtual device.

- Transformation (N:M). Physical IT resources can be transformed from a number of N to M by first aggregating N physical nodes with a technology like vSMP and then virtualising the resulting node using for example ScaleMP. This allows KVM and Xen virtual machines to be run on top of a giant vSMP virtual machine. Regarding storage, a storage pool consisting of N physical disks can be partitioned into M logical storage units, combining 1:N and N:1 paradigms, using for example SNIA technology.

2.9.4.5 *Management of virtualised infrastructure*

The LICL offers a set of tools to a VIP in order to compose and manage virtual infrastructures from a range of abstracted resources coming from different physical domains. The virtual infrastructure composition functionality enables the VIP to offer virtual infrastructures to the different Virtual Infrastructure Operators (VIOs) operating the virtual infrastructure. It enables any kind of virtual resource to be attached to or detached from a virtual infrastructure. On the other hand, virtual infrastructure management comprises a set of functionalities that guarantee coherence and consistency within the LICL. Management capabilities also allow the enhanced GEYSERS NCP and VITM (or a proprietary Network Management System (NMS)) to control the virtual resources.

The LICL management mechanisms also allow dynamic re-planning of virtual infrastructures, offering the VIO the capability to automatically request new resources or even release unused resources during the operational stage of the virtual infrastructure (through the NCP or the SML). The NCP-LICL Interface (NLI) allows the NCP to request the creation/modification of virtual nodes and virtual links during dynamic VI re-planning.

As a Virtual Infrastructure (VI) is a collection of Virtual Resources (VRs), the LICL offers management functions at VI level or at VR level. At VR level, the different types of VR have different sets of management functions. Examples of the types of resources under LICL management are virtual machines, storage and network resources. The VIP itself can use these management functions in order to optimise the utilisation of its resources but they are also provided to the VIO as operations. At VI level, VI-wide management operation can be performed through the LICL and provided by the LICL to the upper layers. They are of two types: batch VR management operations, consisting of VR management operations applied to a set of the VRs constituting a VI, and complex management operations, providing coordinated management operations such as the migration of a part of the VI, or VI duplication.

The virtual infrastructure management operations are supported by a complex security framework. The GEYSERS security infrastructure is proposed according to standard recommendations, best practices and the previous experience of project partners.

The main functionalities of security services in GEYSERS security infrastructure at the LICL are as follows:

- Access control services: for the operation of multiple layers in a distributed environment and to protect VIs and VRs after delivering to their users tenants.
- Policy management and enforcement: for managing resources across multiple domains. It assures consistent, unambiguous policies for resources in a heterogeneous environment.
- Dynamic trust model implementing trust relationships between VRs and the VIO and among VRs within a VI instance. This mechanism, along with access control, forms the basis for data confidentiality and integrity functionalities.
- Secure session context management at LICL for resources during their lifetime. This functionality defines security information formats for resource contexts, sharing security resource contexts to components in the cross-layer architecture.
- Security services for data of VRs, VIs, Physical Resources (PRs) and communications at LICL interfaces. These services can be built using existing security mechanisms that have been proved to be safe, such as encipherment, digital signature, integrity, etc. The security services for LICL support only key establishment and management uses for these mechanisms. Transport and message layer security in LICL inter-services communication can be achieved with the standard transport and message layer security mechanisms such as WS-Security, XML-Security, SSL/TLS, HTTPS, IPSec that are typically available as standard libraries as part of modern network control and management platforms. They can call from LICL services/interfaces using the standard GSS-API.

2.9.4.6 *Control of virtualised infrastructure*

When the LICL is operated over a physical infrastructure, the outcome is multiple isolated virtual infrastructures. An instance of an NCP or a VITM has to be able to operate over each virtual infrastructure for the control and provisioning of its virtual resources. Therefore, the LICL provides a set of tools and mechanisms for each virtual infrastructure to allow the NCP and VITM to operate on the VI through an API. When the LICL creates a virtual infrastructure, it also generates the Virtual Infrastructure Management System (VIMS). The VIMS will be used by the NCP and VITM for the control and provisioning of virtual resources over a virtual infrastructure.

The interface between the NCP and the LICL for control operations on virtual infrastructure is called the Connection Controller Interface (CCI). It is used for the configuration and monitoring of the virtual network resources at a specific virtual network node during service provisioning, and runs between a GMPLS controller and the VIMS handling the associated instance of virtual node.

The CCI supports the following functionalities:

- Virtual network resource synchronisation.
- Support for energy-related parameters (e.g. power consumption, adopted technologies, environmental impact indicators).
- Virtual network resource configuration.
- Support for advance reservation.
- Virtual network resource monitoring and notifications.
- Support of Authentication and Authorisation (AA) in coordination with LICL.

The SML to LICL Interface (SLI) is an interface between SML and LICL that is responsible for the control and configuration of virtual IT resources and for the re-planning of the Virtual IT infrastructure. While the SML is responsible for translating high-level service provisioning requests into technical, executable service provisioning action invocations, the LICL is responsible for receiving and coordinating these invocations amongst the heterogeneous virtual infrastructure components.

Security and access control mechanisms are provided that ensure secure control operations over virtual infrastructures.

2.9.4.7 *Implementation of user interface*

Two user interfaces can be identified in the GEYSERS architecture: the SML and the Network + IT Provisioning Service User-Network Interface (NIPS UNI).

The SML exposes an interface to application providers and customers, such that the complexity of network and IT provisioning is transparent to them. All service requests from the application side will be handled by the SML and translated to technology-specific requests before the provisioning of services over the Virtual Infrastructure. Business objectives for a specific application scenario are declared to the SML and translated into provisioning requests understood by a Virtual IT Manager. The Virtual IT Manager is in charge of the end-to-end IT service management and the virtual IT resources configuration.

This user interface is also used to send virtual infrastructure creation requests generated by applications or consumers.

The application's/consumer's provisioning requests are passed to the NCP by means of the Network + IT Provisioning Service User-Network Interface (NIPS UNI). It allows the cooperation of SML and NCP for the coordinated on-demand provisioning of network and IT resources. It supports multiple functionalities for the NIPS management, including requests for service setup, tear-down, modification and monitoring mechanisms.

The NIPS UNI supports the following functionalities:

- Advertisement of IT resources availability – processing, storage, memory and digital.
    - Support for power-consumption parameters.

- On-demand setup and tear-down of network services in support of IT services.
  - Unicast, assisted unicast, restricted and full anycast connections.
  - Support for advance reservations.
  - Support for QoS constraints.
  - Support for authentication and authorisation procedures.
- On-demand modification of pre-established network services.
- Network and Network + IT service monitoring and notifications.
  - Support for cross-layer service recovery.

2.9.5 Multi-domain support

The GEYSERS architecture natively supports multi-domain environments. However, the support is considered at different levels:

1. Physical Infrastructure Providers (PIPs) offer their physical equipment for a composition of virtual infrastructures. The PIP offers resources to Virtual Infrastructure Providers (VIPs). A single VIP may handle a number of PIPs with their infrastructures. It is the VIP who composes a virtual infrastructure to be spread among a number of administrative domains.
2. The VIP offers virtual infrastructures to Virtual Infrastructure Operators (VIOs). The VIO runs specific services on top of this VI. From the architectural point of view, it is possible that two different VIOs, operating different VIs, are expected to interconnect. This situation requires special mechanisms for exchanging knowledge of network and IT topology and other relevant information to form a new specific service spanning this multi-domain environment.
3. Once a VI is created, the VIO runs a dedicated control plane to allow an instantiation of advanced network services in the network. It is expected the VIO will partition the VI into a number of routing domains to optimise the configuration of the control plane on top of this virtual infrastructure. At the same time, in order to enable multi-technology in a single administrative domain, the VIO partitions its VI into multiple technology domains.

2.9.6 Testbed implementation and availability

GEYSERS plans to deploy a European-wide optical network testbed based on the existing infrastructures, e.g. from the Phosphorus FP6 project and other national initiatives interconnected with GÉANT and GLIF networks. These local infrastructures will offer optical switching access to high-performance IT facilities and network-based IT resources. The GEYSERS testbed will be used for the deployment, validation and demonstration of GEYSERS outcomes in real distributed optical infrastructure applications.

The deployment of the GEYSERS testbed is in progress. The first release of the testbed is expected to be in September 2012.

2.9.7 Current status and roadmap

GEYSERS is at the peak of the third and final year of the project. At this stage, the developments of the GEYSERS software prototypes are reflecting all the design and studies performed in the first two years, which mainly involve the architecture layering reference model design and definition and the studies of GEYSERS business models. The software implementation progress has already started integration activities towards its final deployment in the test-bed provided by the GEYSERS partners and interconnected through GEANT. The final integration and prototypes validation is expected to happen during the third quarter of the year while the fourth quarter will be devoted to demonstrations and dissemination of GEYSERS final product and results.

2.9.8 References

[GEYSERS-D2.1]	GEYSERS deliverable D2.1 “Initial GEYSERS Architecture and Interfaces Specification” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_2.1.pdf
[GEYSERS-D3.1]	GEYSERS deliverable D3.1 “Functional Description of the Logical Infrastructure Composition Layer (LICL)” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_3.1.pdf
[GEYSERS-D4.1]	GEYSERS deliverable D4.1 “GMPLS+/PCE+ Control Plane Architecture” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_4.1.pdf

2.10 NOVI

The information in this section is based on two NOVI deliverables: “D3.1 State-of-the-Art Management Planes” [NOVI-D3.1] and “D4.2: Use Cases” [NOVI-D4.2].

2.10.1 Introduction

Networking innovations Over Virtualised Infrastructures’ (NOVI’s) vision stems from the acknowledgement that computing and network infrastructure developments and virtualisation are rapidly changing the data communication and computation environment. The legacy protocols and standards in these areas need to be revised and extended beyond their original scope. This leap, in effect a paradigm shift, has to cope with the high speed of transition towards the Future Internet (FI) as a comprehensive ICT cloud of composite services. NOVI takes a research and engineering approach. Its objectives are to investigate and experiment on open questions on monitoring, formal description and brokerage of virtualised resources within a federation of FI platforms.

Resources belonging to various levels, i.e. networking, storage and processing, are in principle managed by separate yet interworking providers. NOVI will concentrate on methods, algorithms and information systems that will enable users to work within enriched isolated slices, baskets of virtualised resources and services provided by federated infrastructures.

NOVI will investigate federation at the data, control, monitoring and provisioning planes of constituent FI infrastructures. A user ideally expects seamless and secure access to resources distributed across multiple domains. The complex multi-domain nature of the federated infrastructure requires adoption of common definitions and abstractions of virtualised resources. Users should be able to efficiently identify and correlate virtual resources with desirable attributes and states, while providers should be required to export abstracted views of their offerings, as dictated by scalability constraints and operational concerns. Within this context, NOVI will propose and test resource description data models and abstraction algorithms, incorporating Semantic Web concepts.

Access control is another key issue in federated environments. Authentication and Authorisation Infrastructure (AAI) for user access is an area in which several architectures are being deployed, e.g. the federated schema developed within the NREN world. Secure, authenticated access mechanisms need to transcend protocols and descriptions of virtualised resources of FI federations, the scope of NOVI. NOVI in its experimental phase will investigate options of federated AAIs as they fit its objectives.

Cloud end users are expected to be the administrative owners of their slices, empowered by the ability to configure virtual networking interfaces, protocols and/or the distributed processing resources of the complex holistic FI environment. They should have full access to their slice, including the right to upload/configure monitoring tools, while having restricted access to data from passive and active monitors of the general infrastructure. NOVI's resource allocation algorithms will enable them to dynamically seek resources and negotiate with federated management centres for slices with QoS guarantees, to obtain services of predictable and deterministic behaviour. In FI research, this translates to the ability to plan reproducible experiments over virtualised complex testbeds, such as the federated Future Internet Research and Experimentation (FIRE) facility and the Global Environment for Network Innovation (GENI) experimental platforms. The NOVI consortium will assess the effect on reproducibility of monitoring and brokerage methods as applied in virtualised clouds exported by complex, inter-domain infrastructures. It is expected that on appropriate e-infrastructure substrates, allocation of virtualisation instances, interconnected via virtual switches and logical routers, can lead towards predictable measurable services. In addition, long-haul core connectivity may use over-provisioned substrates, such as European NRENs and GÉANT, which implement a protocol suite richer than what is widely available through the legacy Internet.

The effort will be conducted with a rapid prototyping cycle, using the cutting-edge experimental FIRE facility. More specifically, the proof-of-concept phase will primarily rely on federating resources of the PlanetLab [PlanetLab] Europe and FEDERICA [FEDERICA-DSA1.1] testbeds. End users of this phase will be selected amongst NOVI participants (network research laboratories), acting as guinea pigs to promote adoption by the wider FI community and to substantiate input to standardisation bodies. It is expected that some of the models and methods developed within NOVI will be used to enrich the FIRE facility, in effect contributing to the creation of a blueprint of FI federated infrastructures.

In summary, the specific research goals of the NOVI Specific Targeted Research Project (STREP) concentrate on:

- How to federate different kinds of resources in virtualised e-infrastructures.
- How to formally describe virtualised network and cloud objects in a complex environment, assisted by semantic methods. What ontologies are best suited to describe resources of different kinds.

- How to build slices of virtualised infrastructure at the data, control, monitoring and provisioning planes. How to describe their relationships and technical attributes.
- How to (co-)allocate resources with QoS attributes and how to set up the monitoring system to allow for accountable, predictable Future Internet services.
- How to enrich the FIRE facility with federated models and methods enabling comprehensive and reproducible experiments.

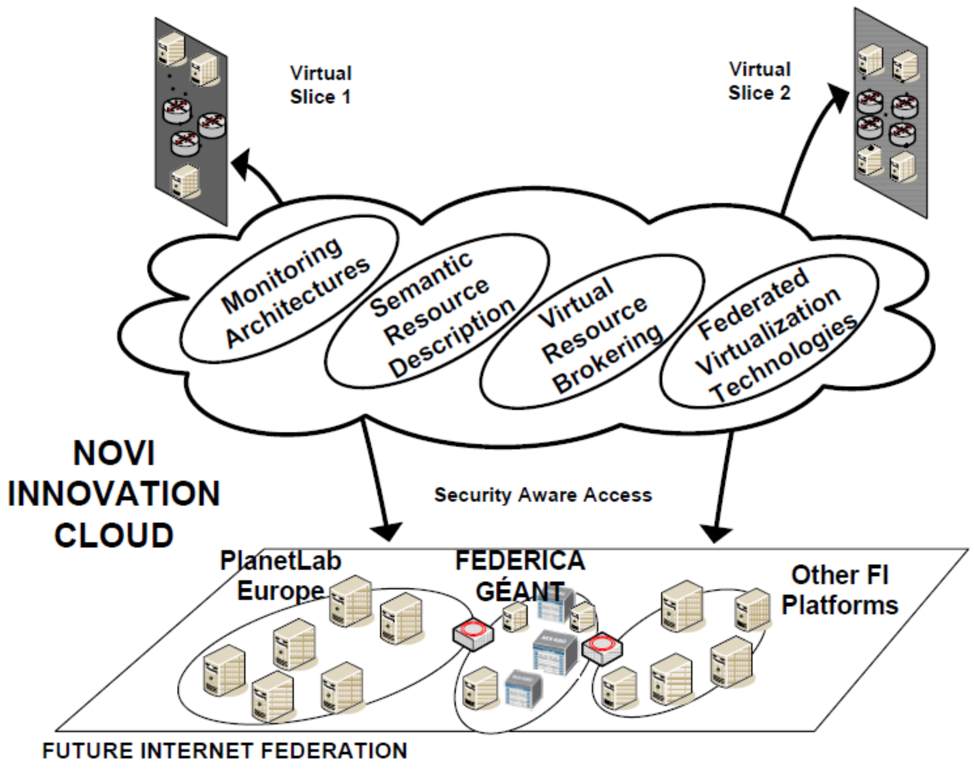


Figure 2.8: NOVI Innovation Cloud

2.10.2 Architecture overview

This information is taken from Section 4.2 of NOVI deliverable “D3.1 State-of-the-Art Management Planes” [NOVI-D3.1].

A preliminary view of NOVI’s Control and Management Plane is shown in Figure 2.9. This is only a conceptual view of its capabilities in terms of its APIs. Implementation of prototypes of NOVI Control and Management API aims at enhancing federation approaches such as Slice Federation Architecture (SFA) and Teagle with advanced services to facilitate slice control and management within a federation of heterogeneous virtualised infrastructures.

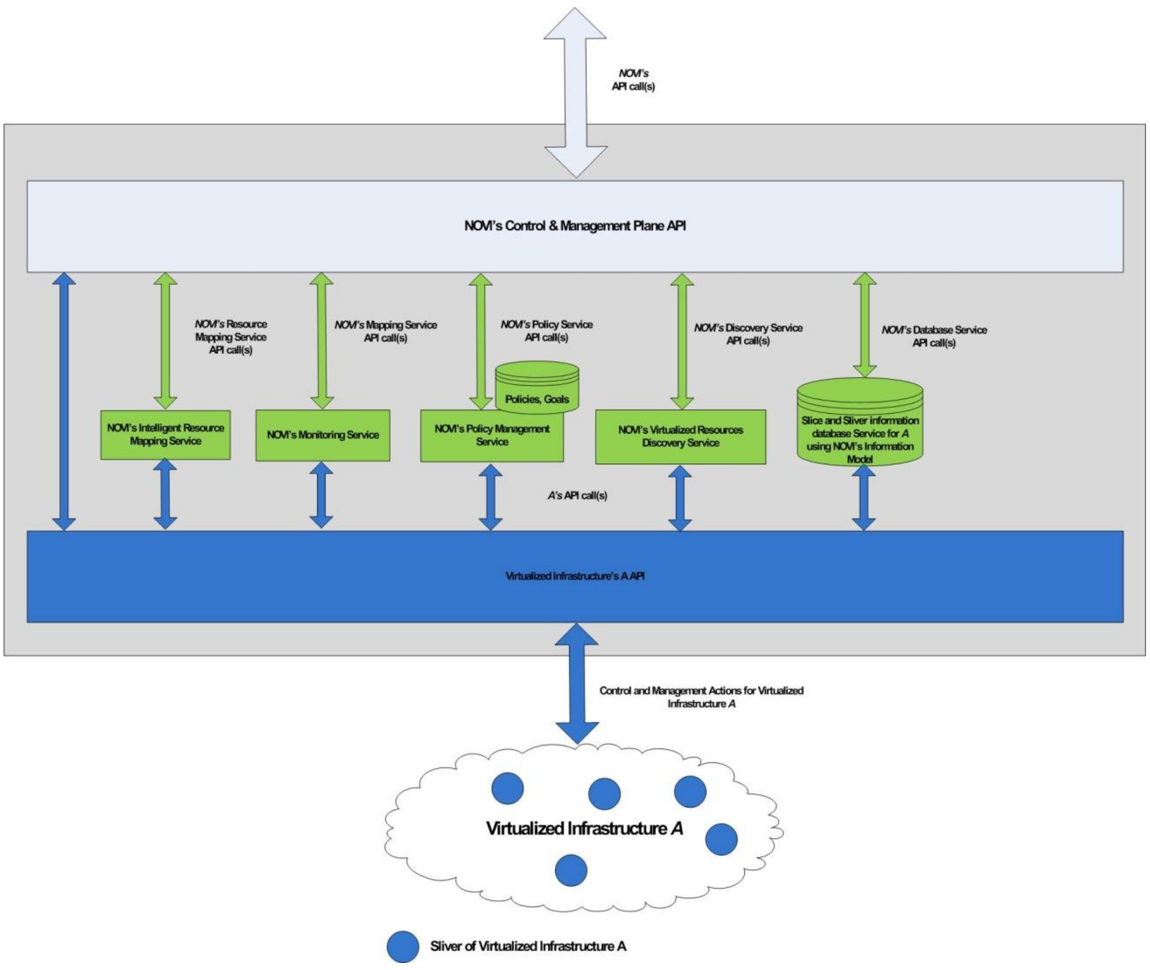

NOVI's API call(s)
NOVI's Control & Management Plane API
NOVI's Resource Mapping Service API call(s)
NOVI's Mapping Service API call(s)
NOVI's Policy Service API call(s)
NOVI's Discovery Service API call(s)
NOVI's Database Service API call(s)
Policies, Goals
NOVI's Intelligent Resource Mapping Service
NOVI's Monitoring Service
NOVI's Policy Management Service
NOVI's Virtualized Resources Discovery Service
Slice and Sliver information database Service for A using NOVI's Information Model
A's API call(s)
Virtualized Infrastructure's A API
Control and Management Actions for Virtualized Infrastructure A
Virtualized Infrastructure A
Sliver of Virtualized Infrastructure A

Figure 2.9: NOVI's Control and Management Plane functionality: a preliminary conceptual view

As shown in Figure 2.9, NOVI plans to offer a NOVI API as a combination of the virtualised platform API and the one developed for accessing novel NOVI Services, depicted as rectangles in Figure 2.9. These act as control and management services and may need to interact in order to decide which management and control actions need to be enforced within the managed environment. When control and management actions are determined, networking federation approaches such as SFA or Teagle could be used to enforce them either within the underlying virtualised infrastructure. NOVI's initial work considers the implementation of methods providing functionality to the following NOVI Services:

- An Intelligent Resource Mapping Service that provides the functionality to support Virtual Network Embedding (VNE) of user requests within the physical substrate. Efficient sharing of virtualised infrastructures requires techniques for solving the VNE problem. VNE provides a mapping of user requests to specific substrate nodes and links. The Intelligent Resource Mapping Engine may consider various alternatives for solving the VNE problem. For example, requests concerning individual resources – slivers – may lead to the adoption of a greedy node-mapping algorithm, whereas user requests for baskets of resources – slices – require solving the full VNE problem via appropriate heuristic algorithms.

- A Discovery Service is able efficiently to find resources based on their context, i.e. “find a computer resource that has CPU utilisation less than 30% and is within the remote domain”. Peer-to-peer algorithms could be exploited to implement the Resource Discovery to cater for a large-scale environment consisting of many resource providers, i.e. authorities offering their resources within a federation.
- A Monitoring Service enables NOVI users and administrators to retrieve information about the temporal behaviour of the status evolution of specific resources that provide such information, and of the network substrate via active network measurements. It is important that the Monitoring Services within NOVI provide semantics-aware information. The output of monitoring calls can support Provisioning Services with dynamic information to find the proper solution for a user’s resource request.
- A Policy Service is used to provide the functionality of a policy-based management system, where policies are used to define the behaviour governing the managed environment. NOVI intends to provide support for event-condition-action policies that enforce control and management actions upon certain events within the managed environment, re-enforcing calls on other NOVI services with different parameters. For example, an event-condition-action policy rule may re-trigger the Intelligent Resource Mapping Service to find a new solution to the VNE if the network graph changes at run-time upon failures or congestion. NOVI also plans to provide support for role-based access control policies which could be used to define different classes of users, receiving different usage priorities on specific virtualised resources.
- A Database Service holds information on slices and slivers within the managed virtualised infrastructure. Virtualised resources are described based on the semantics of NOVI’s Information Model.
- Other NOVI services. These will be defined as an outcome of the Spiral Methodology that will be followed by NOVI, where at the end of the first prototype development cycle (iteration or spiral), various alternatives of NOVI required services to be integrated within the final NOVI prototype will be validated.

Figure 2.10 presents a conceptual view of NOVI’s Control and Management functionality in a simple federation scenario, in terms of how NOVI APIs could be used to communicate control and management information between two heterogeneous virtualised infrastructures. In Figure 2.10, both virtualised infrastructures (platforms) A and B provide an external interface to each other, in terms of secure API calls within the federation. This provides the functionality for one platform to call a remote method in another platform, using secure API calls.

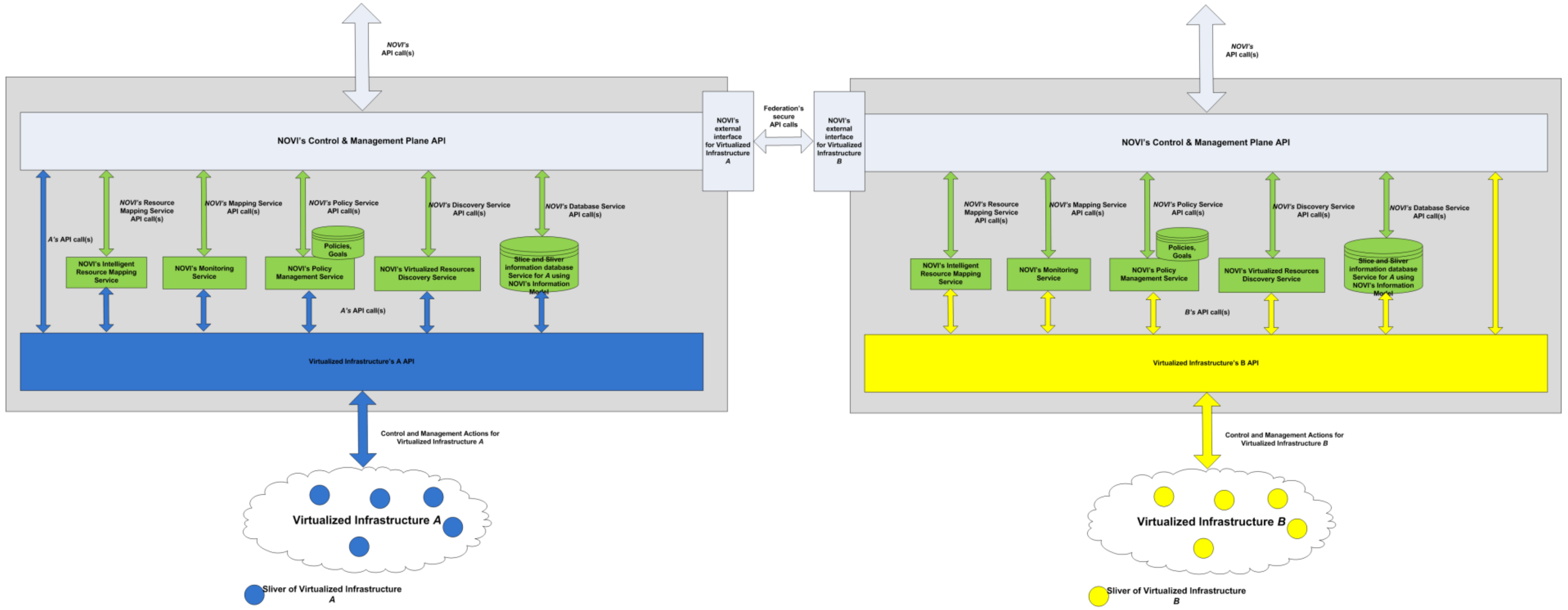

NOVI's API call(s)
NOVI's Control & Management Plane API
NOVI's external interface for Virtualized Infrastructure A
Federation's secure API calls
NOVI's external interface for Virtualized Infrastructure B
A's API call(s)
NOVI's Resource Mapping Service API call(s)
NOVI's Mapping Service API call(s)
NOVI's Policy Service API call(s)
NOVI's Discovery Service API call(s)
NOVI's Database Service API call(s)
Policies, Goals
NOVI's Intelligent Resource Mapping Service
NOVI's Monitoring Service
NOVI's Policy Management Service
NOVI's Virtualized Resources Discovery Service
Slice and Sliver information database Service for A using NOVI's Information Model
A's API call(s)
Virtualized Infrastructure's A API
Control and Management Actions for Virtualized Infrastructure A
Virtualized Infrastructure A
Sliver of Virtualized Infrastructure A
NOVI's API call(s)
NOVI's Control & Management Plane API
B's API call(s)
Virtualized Infrastructure's B API
Control and Management Actions for Virtualized Infrastructure B
Virtualized Infrastructure B
Sliver of Virtualized Infrastructure B

Figure 2.10: NOVI's Control and Management functionality in a federation scenario: a preliminary conceptual view

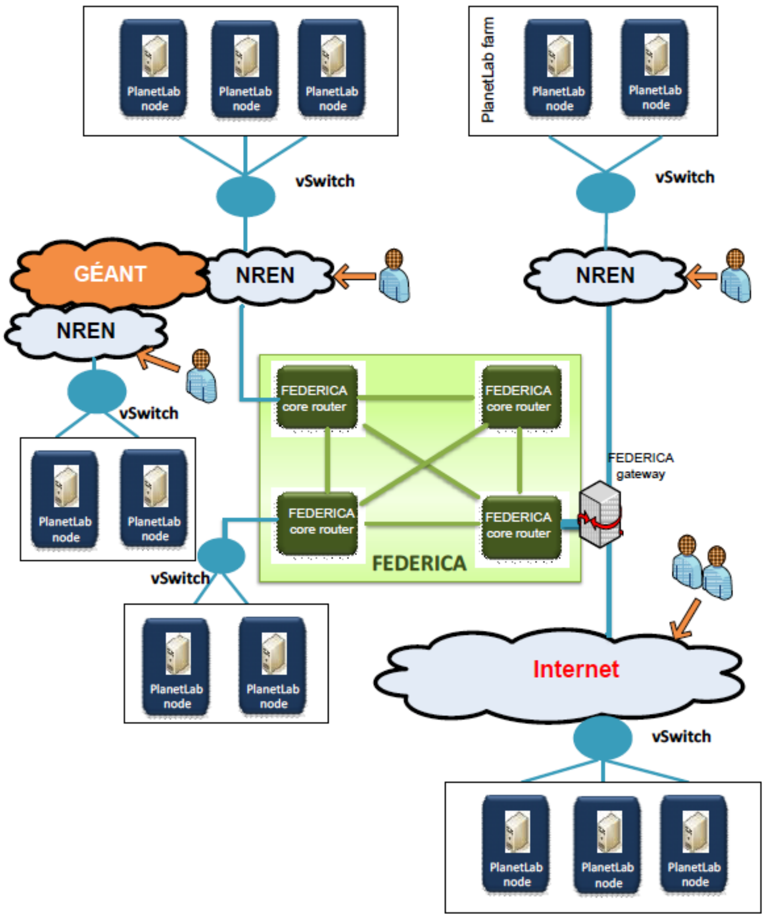


Figure 2.11: FIRE federated environment tailored to be used for NOVI experiments

As shown in Figure 2.11, external users will gain access to the federated testbed in two ways [NOVI-D3.1, NOVI-D4.2]:

1. Via their campus connection to a host NREN and GÉANT for inter-NREN connectivity.
2. Via the public Internet.

In all cases, an authentication mechanism will be implemented within NOVI that will federate PlanetLab and FEDERICA user authentication methods and credentials. Note that FEDERICA users currently log into a Gateway as a proxy to the FEDERICA infrastructure, while PlanetLab users are authenticated by the PlanetLab Europe federated access control.

In order to establish links between PlanetLab Europe and FEDERICA resources, a number of Virtual Switches (vSwitches) will be designed and implemented within NOVI. These will operate at Layer 2, among virtual machines, interconnecting their virtual Network Interface Cards (vNICs). Thus, different virtual networks can operate in parallel, sharing the same physical resources but being isolated at the link layer. As a result, NOVI will enable provision of extended slices as a service, with one slice assigned to PlanetLab Europe and the other to FEDERICA.

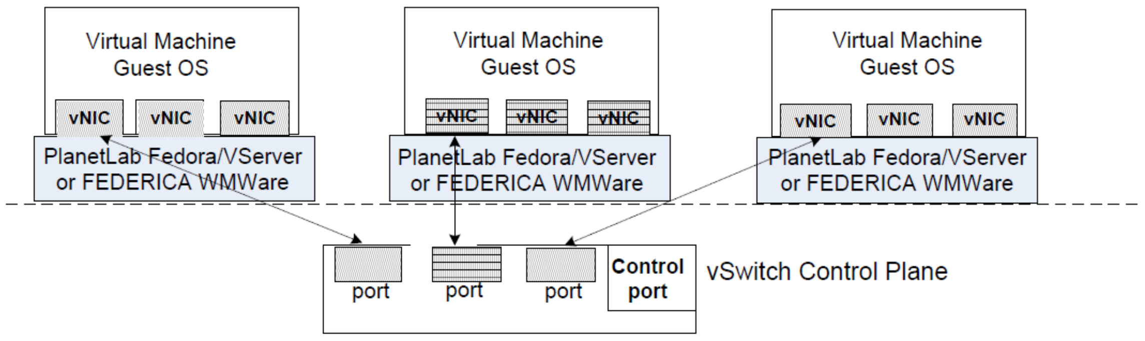


Figure 2.12: vSwitch high-level view

In a virtualised environment, Virtual Machines (VMs) are allocated to users, each running an Operating System referred to as Guest OS. The software layer providing the virtualisation is either hosted by a Host OS (Fedora/VServer in PlanetLab) or runs on bare hardware (ESX VMware [VMWARE] in FEDERICA).

In the FEDERICA world, a slice is realised within the data plane. In contrast, PlanetLab only enables deployment of slices at the application layer, as an overlay deployed on top of the legacy Internet. Interconnection of VMs between PlanetLab and FEDERICA adheres to the different technology layers of the two FIRE facilities. It will be supported via NOVI's specific design and implementation of the vSwitch, as illustrated in Figure 2.12.

2.10.3 User community

There is no user community behind NOVI. NOVI's goal is to exploit infrastructure providers' physical resources.

2.10.4 Mechanisms for providing virtualisation

Work in this area is in progress and not available yet.

2.10.5 Multi-domain support

The future NOVI architecture will support multi-domain environments. The federated testbeds offer their physical equipment for the composition of slices. A single slice may consist of a number of virtualised resources. It is the NOVI middleware that composes a slice to be spread among a number of administrative domains. NOVI offers slices to users. A user runs specific services on top of the slice.

2.10.6 Testbed implementation and availability

NOVI plans to deploy a European-wide optical network testbed based on existing infrastructures, i.e. from the FEDERICA project and the private PlanetLab deployment. The NOVI testbed will be used for the deployment, validation and demonstration of NOVI outcomes.

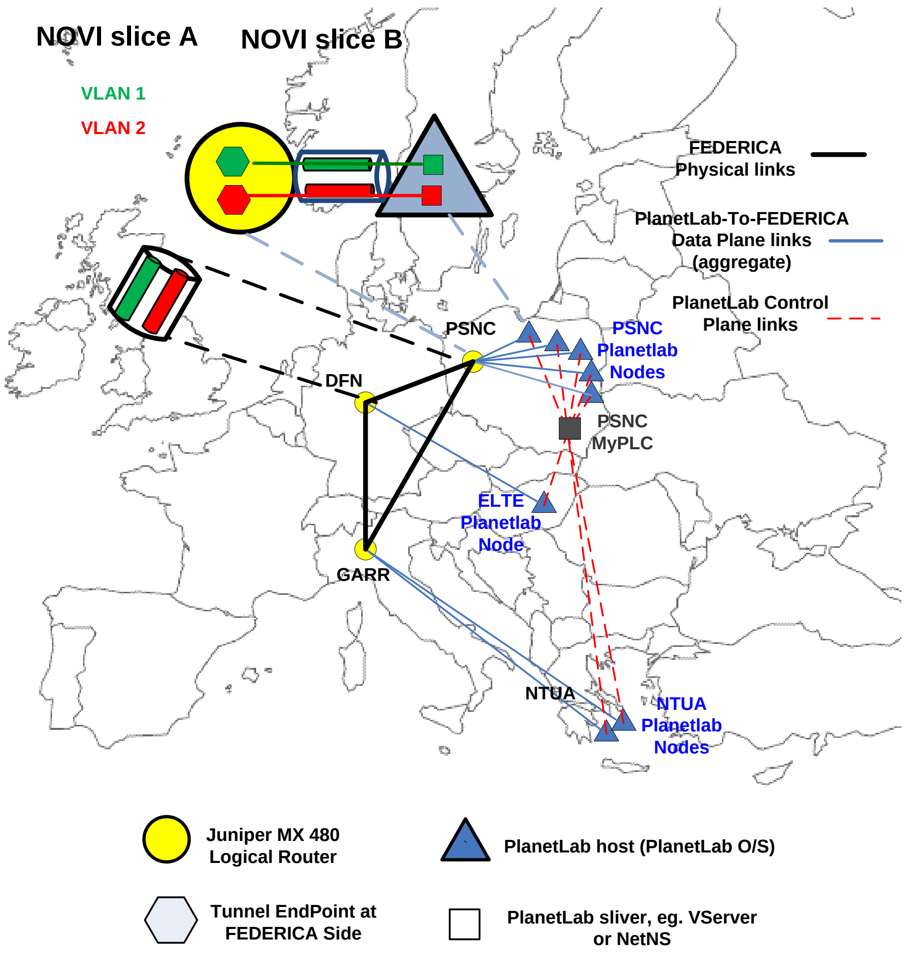

NOVI slice A
NOVI slice B
VLAN 1
VLAN 2
FEDERICA Physical links
PlanetLab-To-FEDERICA Data Plane links (aggregate)
PlanetLab Control Plane links
PSNC
PSNC Planetlab Nodes
DFN
PSNC MyPLC
ELTE Planetlab Node
GARR
NTUA
NTUA Planetlab Nodes
Juniper MX 480 Logical Router
PlanetLab host (PlanetLab O/S)
Tunnel EndPoint at FEDERICA Side
PlanetLab sliver, eg. VServer or NetNS

Figure 2.13: Topology overview

The interconnection of the PlanetLab Europe and FEDERICA infrastructures will evolve in a phased approach. The phased approach will cover evolution issues in the data plane and management-control plane connectivity.

The baseline scenario will verify the possibility of interconnecting the infrastructure without online involvement of the management planes of PlanetLab and FEDERICA. According to this scenario, three core FEDERICA points of presence (PoPs) (PSNC, DFN, GARR) will be set up with a slice involving logical routers cloned from their physical representatives, i.e. the Juniper MX 480 routers. This slice will engage PlanetLab virtualisation functionality i.e. VServer [VSERVER] from remote sites at ELTE, NTUA and PSNC managed by the MyPLC installed on a PSNC server. The data plane connectivity between the VServer and the logical router will be implemented using GRE tunnels.

The Authentication and Authorisation functionality will be done separately, i.e. the user should be registered twice in both infrastructures. The end user will get access to nodes by secure sessions. This simple scenario assumes manual configuration and may not be suitable for large-scale environments. Thus, additional scenarios will be considered.

The deployment of the NOVI testbed is in progress. The first release of the testbed was at May 2011.

2.10.7 References

[FEDERICA-DSA1.1]	FEDERICA deliverable "DSA1.1: FEDERICA Infrastructure" http://www.fp7-federica.eu/documents/FEDERICA-DSA1.1.pdf
[NOVI]	http://www.fp7-novi.eu/
[NOVI-D3.1]	NOVI deliverable "D3.1 State-of-the-Art Management Planes" http://www.fp7-novi.eu/index.php/deliverables/doc_download/24-d31
[NOVI-D4.2]	NOVI Deliverable "D4.2: Use Cases" http://www.fp7-novi.eu/index.php/deliverables/doc_download/26-d42
[PlanetLab]	http://www.planet-lab.org/
[VMWARE]	http://www.vmware.com/
[VSERVER]	Virtualization for GNU/Linux systems http://www.linux-vserver.org/

2.11 OFELIA

2.11.1 Introduction

The current Internet is a mix of heterogeneous technologies, and different research has shown that the current underlying Internet architecture is not sufficient to support the emerging applications in the future. In the past there have been many creative ideas in the area of networks which didn't quite make their way into the production networks to produce a better Internet architecture. One of the main reasons why new ideas cannot be tested on production networks is the fear, given the criticality of today's networks, of the downtime to the business, and also the closed support from the vendors. To overcome these obstacles to testing innovative ideas and redesigning the Internet architecture, OpenFlow was developed.

OpenFlow [OpenFlow1] is an initiative by a group of people at Stanford University as part of their clean-slate program to redefine the Internet architecture. The underlying principle of OpenFlow is to treat traffic as flows, either packet-based or circuit-based traffic at different granularity. The idea behind OpenFlow is to have the control functionality taken out of the equipment (i.e. switch, router) and given to a centrally managed or distributed system, while retaining only data plane functionality on the equipment. This concept combines the advantages of the switching speed of the ASICs (Application Specific Integrated Circuits) and computing flexibility of the PC.

The OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) [OFELIA] project is intended to provide an infrastructure facility for conducting Future Internet experiments using OpenFlow technology. The OFELIA infrastructure facility consists of five different islands spread across the Europe. Each island will host different capabilities to offer different functionalities to the researchers.

OFELIA is funded by the European Union as part of its FP7 ICT work programme. The OFELIA project consortium is made up several academic partners, commercial organisations (including NEC, ADVA Optical Networking) and telecom operators. It began in October 2010 and it completed its first phase with an initial setup in every. It is currently due to finish its second phase by interconnecting islands for enabling multi domain test and experiments. The project is due to run until September 2013 and is expected to continue with new partners joining the consortium.

2.11.2 Architecture overview

At the time of writing this document, OFELIA architecture is still under development. However the architecture will be based on OpenFlow technology. A network is managed by a network-wide operating system running on top of a controller (NOX) [NOX] which controls the data plane of the OpenFlow-enabled network equipment through the OpenFlow protocol. The OpenFlow controller is a server that has the capabilities to host different network management and control applications to effectively manage the network in a centralised or distributed way. This separation between the control and data plane and the capability to treat packet and circuit traffic as flows make the OpenFlow protocol a single standardised control for both packet and circuit networks. There have been several attempts and proposals to control both circuit-switched and packet-switched networks using the OpenFlow protocol. Figure 2.14 shows the unified architecture OpenFlow provides.

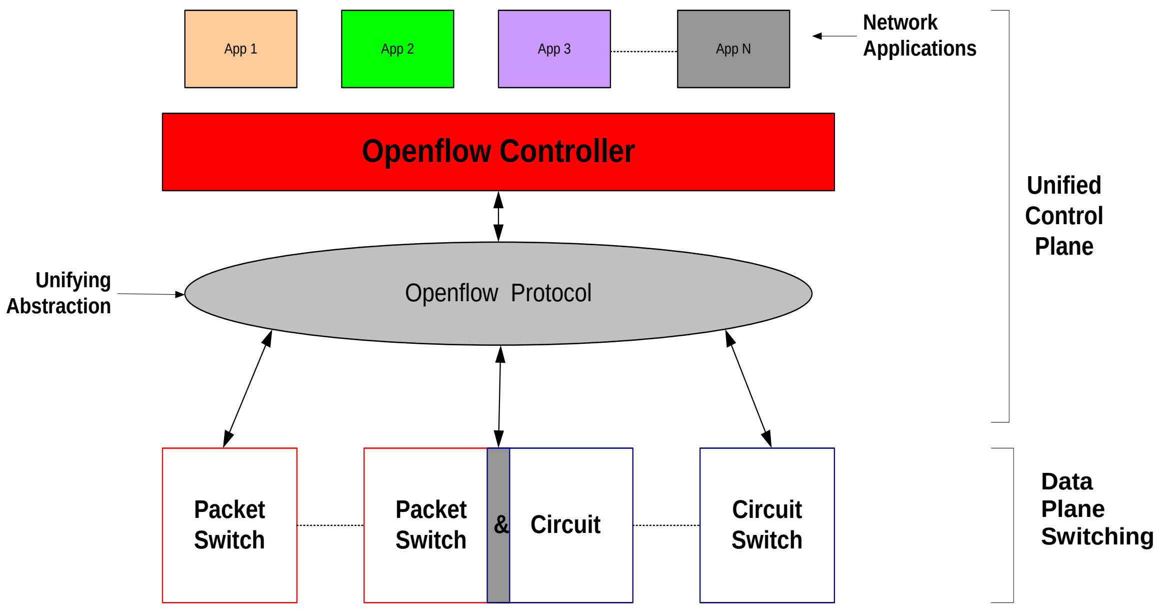


Figure 2.14: OpenFlow architecture

A key component of the OpenFlow architecture is the flow-level virtualisation of the network and its resources. The OpenFlow API provides the programmability ingredients of virtualisation, where clients can program the switches by flexibly defining flow according to their needs and inserting them into the flow tables. The component used for providing virtualisation in an OpenFlow-enabled network is the FlowVisor [FlowVisor2]. FlowVisor is a special-purpose OpenFlow controller which allows the creation of slices from the underlying OpenFlow physical infrastructure. Under the control of a Service Provider it can therefore provide virtualisation isolation in a centralised way. The FlowVisor is housed outside the switch, leaving both the data plane and the controllers untouched. The FlowVisor is transparent both to the switches and to the controllers and it enforces traffic isolation by monitoring and rewriting OpenFlow protocol messages. Therefore, the switches think that they are talking to a single controller, while each controller thinks that it is controlling its own set of OpenFlow-enabled switches. OpenFlow-enabled virtualisation (i.e., FlowVisor) allows the transport service providers (e.g., network operators) to retain control over the transport network, while allowing clients (such as an ISP) to use whatever automated intelligent control algorithms they may desire in their isolated slice of the network as depicted in Figure 2.15.

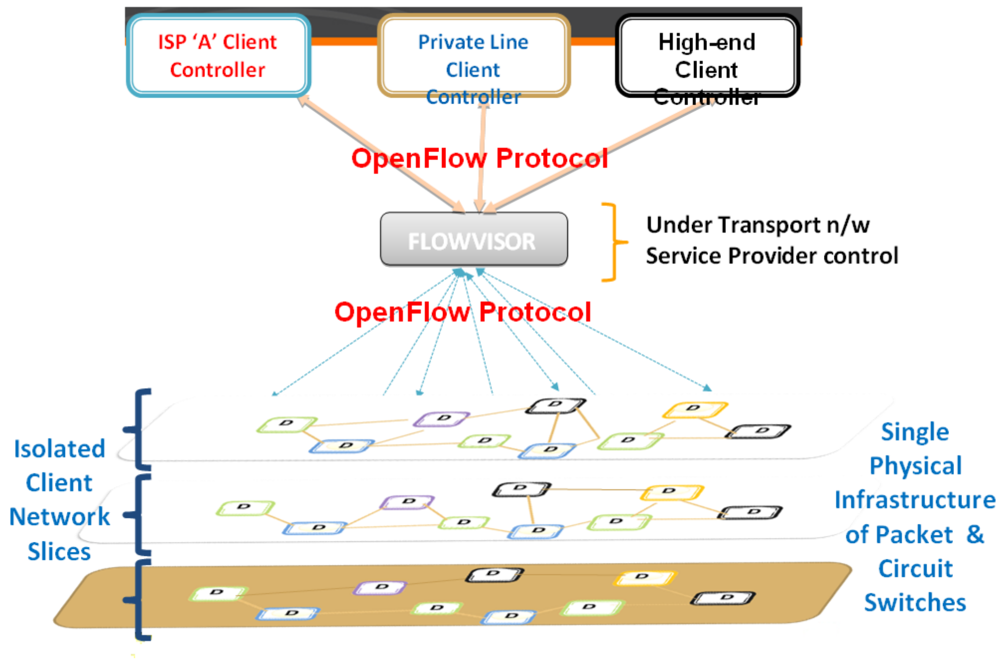

ISP 'A' Client Controller
Private Line Client Controller
High-end Client Controller
OpenFlow Protocol
FLOWVISOR
Under Transport n/w Service Provider control
OpenFlow Protocol
Isolated Client Network Slices
Single Physical Infrastructure of Packet & Circuit Switches

Figure 2.15: OpenFlow virtualisation of physical infrastructure

The transport network resources provided to the Infrastructure Service Provider by the transport service provider can be virtualised further by the ISP for its own needs. This means that it is possible to further virtualise the client network via a FlowVisor, which is under the control of the ISP.

2.11.3 User community

OFELIA aims to provide isolated infrastructure slices (i.e. network + IT resources) for researchers and users who want to deploy, test and evaluate a specific service, network protocol or network management application at a scale of and in parallel with the production networks. OFELIA aims to provide the users with a realistic experimental facility, which emulates the real/production networks, for carrying out the testing of next-generation Internet technologies. However, OFELIA's tools will enable interfacing with many experimental facilities such as PlanetLab and GENI, thus expanding the user community base and the scale of the facility.

2.11.4 Mechanisms for providing virtualisation

OpenFlow aims to devise a new virtualisation technique, which can provide virtualisation of the network as a whole rather than virtualising the network using well-known criteria at different layers in the OSI stack (e.g. L2 VPN, VLAN, L3 VPN, IPv4, MAC/VLANs). The virtualisation can be performed in a very flexible way by using any header field at different layers and not only by using the well-known criteria. Currently, virtualisation of the network using OpenFlow is achieved using a special-purpose OpenFlow controller called FlowVisor as discussed in Section 2.11.2.

2.11.4.1 *Implementation of virtualisation on Layer 3*

At Layer 3, virtualised slices can be created using a specific IP address or IP subnet. Any testing on the existing routing protocols or new routing protocol is written as an application and is run on top of the OpenFlow Controller controlling the slice. No routing computations are done on the network devices. All the routing computations are done on the centralised or distributed OpenFlow controller and the relevant flows are pushed to the OpenFlow-enabled switches.

2.11.4.2 *Implementation of virtualisation on Layer 2*

Virtualisation at Layer 2 can be done by writing policies in the FlowVisor. Layer 2 virtualisation on the FlowVisor can be either VLAN-based or MAC-based.

2.11.4.3 *Implementation of virtualisation on Layer 1*

Within the OFELIA project, a concept of a Layer 1 / Layer 0 slice parallel to a Layer 2 slice should be defined. As Wavelength-Division Multiplexing (WDM) and Ethernet are different techniques, the slice definition is also different. Within the project, different Layer 2 slicing concepts are being discussed (e.g. MAC-based, VLAN-based). For WDM, an intuitive way of slicing is based on resource, which is a wavelength. A difference from Ethernet is that a wavelength is a physical resource and it is not possible to allocate more lambdas than are currently available. Things look different in Ethernet. For the flows it is possible to allocate abstract bandwidth. The total allocated number may be higher than the real capacity of the link thanks to statistical multiplexing of the flows. Two approaches may be considered:

- A slice defined as a fixed set of wavelengths.
- A slice defined as a number of wavelengths available from the whole spectrum.

The latter approach is more flexible and in many cases may provide better network resources utilisation. No implementation is currently available for optical virtualisation. In the later stage of the OFELIA project it will become clearer how virtualisation at Layer 1 / Layer 0 (the optical layer) is implemented in the facility.

2.11.4.4 *Implementation of computing virtualisation*

OFELIA aims to provide an effective mechanism for server virtualisation and isolation such that each part of the server can independently run a specific service and only be part of a specific isolated slice. In the first phase of the project, the server virtualisation support will be provided using Xen hypervisor. The OFELIA control framework will provide interfaces for reserving virtual machines based on Xen hypervisor. It is planned to extend the support for other virtualisation technologies in the later stage of the project.

2.11.4.5 *Control of virtualised infrastructure*

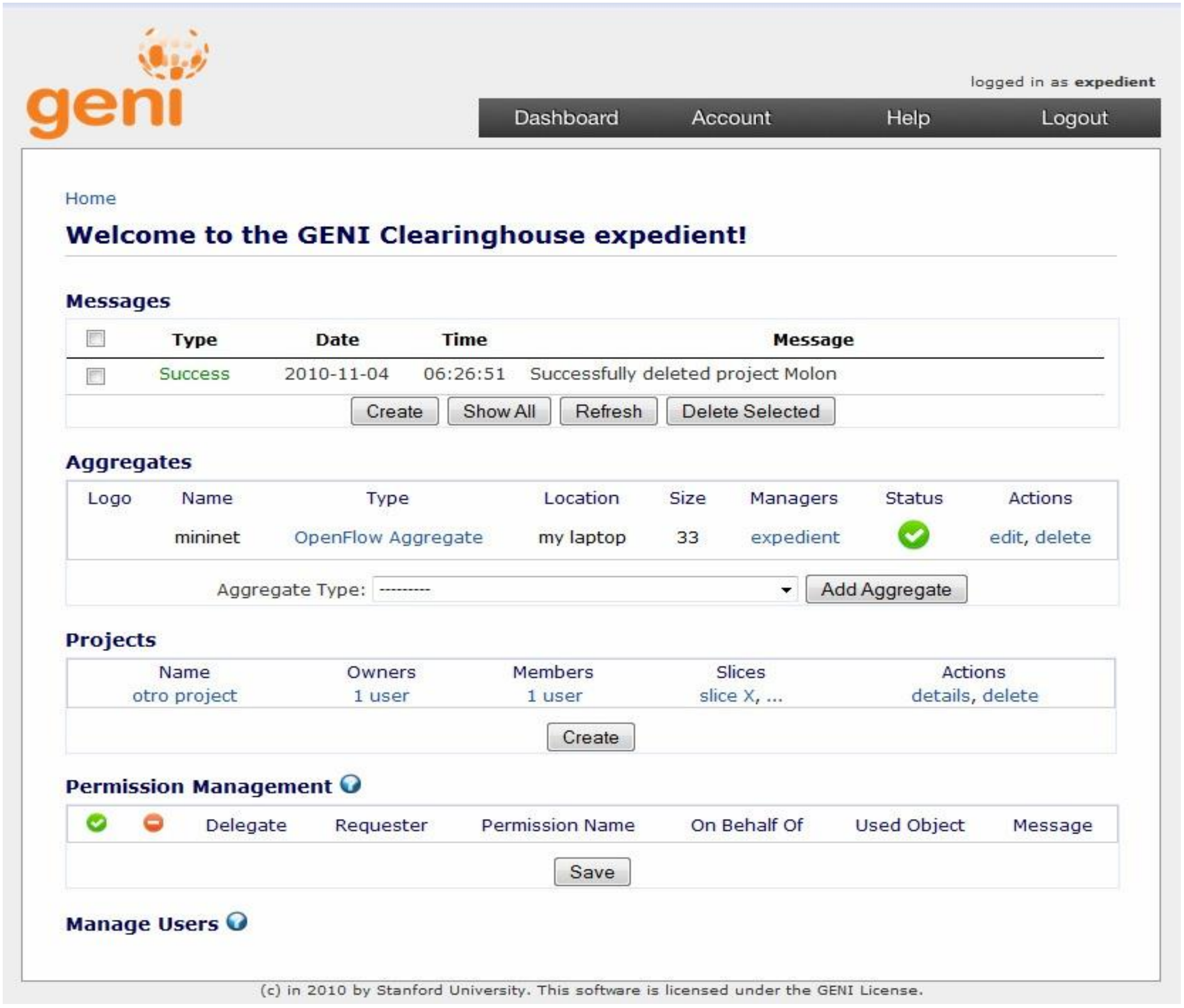
While the underlying physical infrastructure is controlled by the physical infrastructure owner, the users have total control over their slice and are allowed to perform activities within their slice.

2.11.4.6 *Implementation of user interface*

OFELIA will provide a GUI to the end user, which allows them to provision their logical network (slices). The GUI is a researcher portal where the end users register themselves, create and modify the experiments. In OFELIA the researcher portal is Expedient [Expedient], initially developed at Stanford University, used in the GENI OpenFlow campus trials and now being extended to meet OFELIA's needs. The process for reserving a slice is as follows (from the user's point of view):

- Step 1: Register in Expedient.
- Step 2: Register the aggregates (OpenFlow components and VMs) in Expedient.
- Step 3: Create a project.
- Step 4: Create a slice within the project and add aggregates.
- Step 5: Create a Flow Space in the slice, to be approved by the administrator.
- Step 6: After the Flow Space is approved, start the slice.
- Step 7: Carry out the experiments.
- Step 8: Delete the slice after the completion of the experiments.

The user can see the logical, sliced network on the screen after it has been reserved and carry out the experiments. The Expedient dashboard GUI is shown in Figure 2.16.



Home

Welcome to the GENI Clearinghouse expedient!

Messages

	Type	Date	Time	Message
☐	Success	2010-11-04	06:26:51	Successfully deleted project Molon

Aggregates

Logo	Name	Type	Location	Size	Managers	Status	Actions
	mininet	OpenFlow Aggregate	my laptop	33	expedient	✓	edit, delete

Aggregate Type: ---------

Projects

Name	Owners	Members	Slices	Actions
otro project	1 user	1 user	slice X, ...	details, delete

Permission Management

✓	⊖	Delegate	Requester	Permission Name	On Behalf Of	Used Object	Message

Manage Users

(c) in 2010 by Stanford University. This software is licensed under the GENI License.

Figure 2.16: Expedient dashboard

2.11.5 Multi-domain support

OFELIA currently provides a single-domain network. In principle, it is possible to use resources from multiple physical domains using the OpenFlow technology. In the later stages of the project, the OFELIA facility will incorporate the optical and wireless domains. The virtual infrastructure that is created can be multi-domain–based, where researchers can run experiments across different domains.

2.11.6 Test bed implementation and availability

No public test bed is currently available for OFELIA. However, at the end of first phase of the project (June 2011), the testbed became available to external researchers to perform experiments over a L2 network built using OpenFlow-enabled Ethernet switches.

2.11.7 Current status and roadmap

The first implementation of the OFELIA testbed was completed at the end of March 2011, enabling researchers to conduct experiments within an island over a Layer 2 network. In phases II and III of the project there will be integration of optical and wireless support in the facility and experiments can be run across the WAN over the federated OFELIA islands. OFELIA islands will comprise of OpenFlow-enabled network equipment and controllers providing an OpenFlow-enabled network infrastructure along with virtualised server end points to act as source and sinks. In OFELIA, there are five islands across Europe.

2.11.8 References

[Expedient]	http://yuba.stanford.edu/~jnaous/expedient/docs/admin/install.html
[FlowVisor2]	http://openflowswitch.org/wk/index.php/FlowVisor
[NOX]	www.noxrepo.org
[OFELIA]	http://www.fp7-ofelia.eu/
[OpenFlow1]	www.openflow.org

2.12 Google App Engine

2.12.1 Introduction

While most efforts at virtualisation have focused on providing an existing, familiar environment as an abstraction on underlying hardware, Google has taken a different approach. The App Engine is a specific environment, with specific development languages (Python and Java at the time of writing) and abstracted hooks into Google's proprietary architecture for data storage and networking.

This means that the scaling is handled transparently behind the abstraction. Environments based on machine instances, such as VMware virtual machines or Amazon Elastic Compute Cloud (EC2), require the developer to manage the scaling of multiple virtual machines. In Google's environment, by accepting the extra restrictions on the development environment, the scaling is itself abstracted away.

2.12.2 Architecture overview

Google App Engine provides two development environments: Python and Java. This is Google's specific environment; most modules are available but, for example, some Python extension modules written in C cannot be used in the App Engine environment.

Applications run in a secure environment known as the Sandbox, which provides the abstractions to Google's proprietary systems. Direct networking and filesystem operations are not available, and attempts to use those functions in the development languages will raise an exception. Instead:

- For inbound connectivity, application code is run only in response to a web request, queried task, or scheduled event; only http and https inbound connectivity is provided.
- For outbound connectivity, URL fetch and email services are provided.
- Storage is provided by means of the Datastore and Memcache services.

The Datastore is the primary persistent storage for App Engine applications. While there is an SQL-like interface to it, it is not a relational database. Queries are limited in a way that ensures their performance scales with the size of the result set returned, rather than the full data set; for example, every Datastore query must have a pre-built index.

Access to the various services is both policed and billed by a system of quotas. A number of resources are limited specifically to guard against overloading by problematic applications, such as the number of calls that can be made per minute or per day to the Datastore API. Other quotas are set at a certain level for free use, and can be increased for a cost at the developer's request. For example, there is a daily limit of 1 GB outgoing bandwidth for free usage, which can be increased up to 14,400 GB by enabling billing.

Authentication services are provided by means of Google accounts, and OpenID.

2.12.3 User community

Like many of Google's products, App Engine is quite widely accessible, particularly due to the fact that applications can be hosted within the free quotas at zero cost to the user. However, the interface to the infrastructure is via a programming development environment; the users for whom Google is catering are web developers who wish to abstract away the underlying hosting, networking and storage infrastructure that is needed to support web applications.

2.12.4 Mechanisms for providing virtualisation

2.12.4.1 *Implementation of virtualisation on Layer 3*

Google App Engine does not provide networking virtualisation of any sort, except in as much as networking and hosting services are abstracted away from the developer by means of the provided APIs and provided on Google's own infrastructure.

2.12.4.2 *Implementation of virtualisation on Layer 2*

Google App Engine does not provide Layer 2 networking virtualisation.

2.12.4.3 *Implementation of virtualisation on Layer 1*

Google App Engine does not provide Layer 1 networking virtualisation.

2.12.4.4 *Implementation of computing virtualisation*

Google App Engine's primary purpose is to provide computing virtualisation for web applications. All aspects of hosting and networking below Layer 7 are abstracted away from the developer by means of the App Engine's developer APIs, and are provided on Google's proprietary infrastructure.

2.12.4.5 *Management of virtualised infrastructure*

Access to the abstracted infrastructure is managed by means of the APIs, which are moderated by the quotas described above. Usage of the infrastructure can be tracked along several axes, including those quotas, using a control panel that is provided as part of the App Engine service.

2.12.4.6 *Control of virtualised infrastructure*

The underlying virtualised infrastructure is fully abstracted from the developer and user, so no control over the infrastructure is provided except by means of those abstractions. Any attempt to perform low-level networking or filesystem operations fails.

2.12.4.7 *Implementation of user interface*

Google App Engine applications are web applications, so end users generally access them through a web browser.

When developing the applications, developers use a development environment provided by Google that simulates the App Engine environment (including services such as the Datastore) on the developer's own machine. Once the application is working satisfactorily in the development environment, it can be uploaded to the App Engine where it is hosted, either using a free domain under .appspot.com, or using a domain already registered by the developer.

2.12.5 Multi-domain support

Google App Engine does not support multiple domains; it is an interface to Google's own infrastructure.

Because it is a specific and partly proprietary environment, there is the question of how easy it is to port apps in and out of the environment. While there are certainly proprietary aspects to the service, very many of the components are open source and one might expect to implement these easily elsewhere.

When developing an application with a view to porting it out of the App Engine environment later, one would have to pay careful attention to the proprietary services, particularly the Datastore, to ensure that queries can be adjusted to be served by other types of database (such as an SQL-based relational database.)

Porting existing applications that were not targeted at the App Engine may be challenging given the additional restrictions noted above.

2.12.6 Testbed implementation and availability

While there is no separate testbed as such, the App Engine is free to use within certain quotas, and the development environment provides a desktop-based simulation of the environment for development purposes.

2.12.7 Current status and roadmap

Google App Engine is a production service from Google, which attracts both free and paid usage. Ongoing developments are documented in the App Engine Blog [GoogleAEBlog].

2.12.8 References

[GoogleAEBlog]	http://googleappengine.blogspot.com/
[Google-Intro]	http://code.google.com/appengine/docs/whatisgoogleappengine.html

2.13 Amazon Virtualisation

The information about Amazon virtualisation is unchanged. Please refer to [GN3-DJ1.4.1] Section 2.9.1.

2.14 Summary Comparison

Table 2.2 on the following pages provides a summary of the virtualisation technologies described above. More specifically, the following aspects are considered for each virtualisation technology:

- Protocol dependency: states whether there is any protocol dependency for the users of the virtualised infrastructure.
- Network layer virtualisation: the OSI layers for which virtualisation is provided.
- Computing virtualisation: whether computing virtualisation is provided.
- Virtualisation technology: how virtualisation is achieved.
- Reason for deploying virtualisation: what is the added value that virtualisation offers.
- User community: the community that the virtualisation technology is targeting.
- Who manages the virtualised infrastructure. Two broad roles are identified:
    - Physical infrastructure owner – the party that owns the substrate infrastructure that is used for implementing virtualisation.
    - User – the party that exploits the subset of the physical infrastructure that constitutes the virtualised infrastructure.
- Management tools: what are the tools that are used for managing the virtualised infrastructure. It should be specified if these tools are used by the physical infrastructure owner or the user.
- Offered services: the services that are offered to the users.
- Potential use in a multi-domain environment: whether deployment of the virtualisation framework is possible in a multi-domain environment.

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
FEDERICA	None. A user can define its own networking technology	L3/2	Yes	Inherent virtualisation capabilities of L3/2 NEs (Junos and software router/switch) and servers (VMware ESXi)	Creation of parallel virtual environ-ments (slices) aimed at supporting research on networking	Network researchers	Physical infrastructure owner and/or users	Traditional tools. Tools for slice-oriented provisioning, management and monitoring are under development	Creation of L2/L3 VPNs (including virtual computing elements)	Open to be inter-connected/ federated with other e-infrastructure and service management frameworks, e.g. IPsphere
MANTYCHORE	-	L3: configuration of virtual networks, routing protocols, etc. L2: configuration of services for Ethernet and MPLS switches L1: configuration of cards and ports from optical devices	No	Netconf	Ti provide IP networks as a service	Three research user groups: Danish Health Data Network, British Ultra High Definition Media group and the Irish Grid network	Research users	MANTYCHORE GUI	Create links between routers, define IP addresses, define routing protocols	No

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
Phosphorus	IP	L1	No	UCLPv2 based on web service technology	Resource partitioning and network virtualisation through network resource slicing	NRENs and e-science community	Users	Web-based GUI	Static connectivity provisioning Static network topology creation and control Static network slicing	Yes
4WARD	None. The concept is independent of specific protocols	L3	N/A	N/A	Co-existence of multiple architectures and smooth migration path New business models	Addressing all users	Physical infrastructure owner	Implementa-tion of a “Virtualisa-tion Management Interface”	N/A	N/A
GENI	None. A user can define its own networking technology	L3/2/1	Yes	Virtualisation middleware (GENI Management Core – GMC) and inherent virtualisation capabilities of L3/2/1 NEs and servers	Project focus	Network researchers	Physical infrastructure owner	Management tools are under develop-ment. They are accessed by the physical infrastructure owner via the GENI operator	Researchers can define their own experiments over the virtualised infrastructure via the researchers portal	N/A

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
								portal		
PlanetLab/ VINI/OneLab	IP	L3/2	Yes	PlanetLab and VINI virtualisation tools	Infrastructure slicing for protocol testing	Network, application and service researchers but not limited to any community	Slicing and creation of virtual infrastructure in a central authority basis. Management of each slice can be done by users through a dedicated interface	Specific management tool and interface is available	Multiple independent network and server slice over same infrastructure	Yes
AKARI	IP for CoreLab. None for VNode	L3/2	Yes	CoreLab: Planet lab tools with GRE-tap tunnels and virtual OpenFlow switch. VNode: GRE encapsulation, support for MPLS, VLAN, and OpticalPath foreseen (not yet implemented)	Creation of parallel virtual environ-ments (slices) aimed at supporting research on networking	Network researchers	Physical substrate owner and/or users	GUI and XML configuration	VNode allows creation of L2/L3 VPNs, VMs used as routing engines only (no end-nodes)	N/A
GEYSERS	-	L1: optical network virtualisation	Yes	OpenNebula for IT resources	To enable optical network providers to	No user community	Virtual infrastructure providers	Logical Infrastructure Composition Layer (LICL)	Virtual infrastructure composition Management	GEYSERS architecture natively supports

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
					compose logical infrastructure s			management tools	functions at virtual infrastructure level or at virtual resource level	multi-domain environment s
NOVI	Protocol Independent	Layer 2 virtualisation	Imple-mented by Vservers and VMWare VMs	Virtual Machines on VMWare ESXi; Vservers hosted on PlanetLab; Juniper Logical Routers	To enable multiple users to carry out network experiments on their partition (slice) by virtualising the underlying physical infrastructure .	FIRE user communities.	A dedicated management entity within NOVI project.	VMware ESXi Vsphere, MyPLC, various monitoring tools	Web GUI to create virtual network topology; User Access Gateway for using virtual components (virtual nodes and logical routers); L2 federation service (Nswitch); peering service for FEDERICA and PlanetLab (implement-ing SFA); resource monitoring service for	It is designed to offer federation to multi-domain, heterogen-eous infrastruct-ures managed and controlled by different administra-tive bodies (e.g. PlanetLab Europe, FEDERICA)

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
									federated environment; distributed database of resources in federated environment; Intelligent Resource Mapper	
OFELIA	None. Users can define their own protocols.	L1/L2 (ideally OpenFlow tries to flatten the hierarchy of layers)	Yes	Inherent virtualisation capabilities of Xen servers for server virtualisation (end nodes) and OpenFlow-based FlowVisor capabilities for network virtualisation	To enable multiple users to carry out network experiments on their partition (slice) by virtualising the underlying physical infrastructure	Network researchers	Physical infrastructure owners and/or users.	At the moment the web GUI (Expedient) and the management tools are still under development	Researchers can define their own experiments over the virtualised infrastructure via the management portals	Yes. The facility will incorporate wireless And optical in its testbed in the later phase of the project
Google App Engine	App Engine is optimised for web applications. Every inbound request to the app is	App Engine does not provide network layer virtualisation.	Yes	The developer's applications run in a sandbox that almost entirely abstracts away the underlying platform. There are heavy	Cost efficiency. Improved scaling. Avoidance of disk bottlenecks on a given	App Engine is aimed at developers who wish to abstract away the problem of hosting and	The virtualised infrastructure itself is proprietary and managed entirely by	The application's administrator has some (but not complete) control over how the	It is a hosting service for web applications that specifically target its APIs.	The App Engine is a single-domain service, and operates as Platform as a Service

Virtualisation technology	Protocol dependency	Network layer virtualisation	Computing virtualisation	Virtualisation technology	Reason for deploying virtualisation	User community	Who manages the virtualised infra-structure	Management tools	Offered services	Potential use in multi domain environment
	framed as an HTTP request.			restrictions on the application compared to hosting on a traditional VM. For example, there is no persistent filesystem in the application sandbox.	piece of (shared) hardware. Consistent state between instances that are being arbitrarily started and terminated as demand requires.	scaling their applications.	Google. The developer has some control – for example, over the rate at which new instances are spawned in response to demand – but this is limited to the amount of time a request will wait for an existing instance to become free.	application is run. In particular, the administrator can request a minimum latency. It is also possible to directly inspect the contents of the datastore and memcache.		(PaaS) as opposed to the Infrastructure as a Service operations discussed elsewhere in this document.
Amazon	IP	L3	Yes	Amazon specific tool (Amazon Web Services-based virtualisation tool)	Efficient sharing of resources. Increased utilisation of resources	Everyone. Commercial service	Users	CLI, API	Elastic Compute Cloud (EC2) Simple Storage Service (S3) Virtual Private Cloud (VPC)	No

Table 2.2: Summary comparison of virtualisation technologies

3 Drawback Analysis of Virtualisation of Network Services

3.1 Introduction

The developments in Virtual Network Services (VNSs) have attracted significant interest and investment and given rise to many activities around the world. Commercial applications are available, hardware vendors are offering products to implement such services, and within the R&E community many development projects are ongoing. Many of these solutions, services, products and projects are ready to deliver aspects of a technology area that is still developing rapidly. This is true also for the GÉANT community: several NRENs have started or are planning to implement certain solutions for a VNS (for example, HEAnet and NORDUnet are working on the MANTYCHORE project, which offers virtual infrastructure services, and PIONIER, operated by PSNC, is planning to use VNS in future). Within GN3, JRA1 Task 4 is developing an infrastructure virtualisation mechanism for GÉANT, where VNS is the core service the mechanism will provide. The Task has also undertaken a drawback analysis, to identify possible problems or obstacles to implementing a VNS.

This section of the deliverable identifies several areas that could be sources of drawbacks to providing a VNS. Each potential drawback is analysed for its impact on the introduction of a VNS to clients/researchers. The methodology is described in Section 3.2. In Section 3.3 the areas are divided into technical issues, service issues and (more general) provision issues.

A preliminary analysis suggests that the possible drawbacks of virtualisation could be in details that are very sensitive to the transport layer (L1, L2, L3) of the virtual network/virtual service. The layer-dependency covers many areas such as hardware, software, costs, user demands, etc. This study does not separate the analysis according to such specific layer dependency. This might be done in an extra study, if required.

3.2 Drawback analysis methodology

The drawback analysis was conducted to identify risk areas associated with the introduction of Virtual Network Services.

A first analysis shows that the drawbacks can be categorised as follows:

- Technical issues: associated with the requirements and availability of hardware and software.
- Service-oriented issues: concerning the required service aspects seen from the user and the provider point of view.
- Business issues: concerning the costs and the competing services (alternatives).

Each type of drawback is assessed according to the probability that it will materialise as a showstopper and prevent the introduction of a VNS:

- Low: if its probability is lower than 25%.
- Medium: when the probability ranges from 25% to 50%.
- High: if the probability is higher than 50% but lower than 75%.
- Very high: if the probability is more than 75%.

All drawbacks also have a severity level associated with them. This is an indicator of the impact of an actual problem on the introduction of a VNS. In some cases it might also reflect the threshold for the adoption by the end user. The severity is classified as:

- Devastating.
- Serious.
- Medium.
- Tolerable.
- Insignificant.

The term “user” in this analysis refers broadly to GÉANT and NREN users. An initial view of what NREN users might use the virtualisation service for was obtained from the requirements survey, documented in [GN3-DJ1.4.1] Chapter 3.

3.3 Drawback Areas

3.3.1 Technical Issues

3.3.1.1 *Hardware environment*

In principle the hardware required for the provision of a VNS seems to be available (CPU, interfaces, memory, etc.), even within commercial products. It is probable, however, given the limited experience with VNS to date, that some hardware components can still be enhanced/adapted to provide a more effective VNS (e.g. performance, resource isolation), but this is not a major aspect (apart from the possible extra cost) that will hinder the introduction of VNS.

Thus the number of further special requirements, compared with what is already available, is low. The available general-purpose hardware (including updates/upgrades) will mostly fulfil the requirements; sometimes special types of hardware (e.g. line cards) might be required for a good level of VNS, but this is mainly a cost problem.

Drawback analysis:

- Likelihood: Low.
- Severity: Tolerable.

3.3.1.2 *Software environment*

There is a need for a variety of special software to provide, manage and operate the VNS. This special virtual software must be ordered, implemented, operated, and maintained. The amount and quality of this software are strongly linked to the service issues (see Section 3.3.2 below).

The required software is or will be available and it has or will have the required stability for operation, etc. Development is still ongoing and the specific situation for a certain environment must be analysed in detail, especially with regard to the layer of the VNS. Examples of areas of software that still need further development are:

- Resource information and allocation (including limitations of usage).
- Slice/service isolation.
- Missing monitoring features within the virtual environments.

In summary, the required components will be available, but certainly with varying levels of quality (which is always true for software). Insufficient management components will make VNS deployment much harder.

To judge the status, the quality and potential risks of the software more effectively, the drawback analysis distinguishes two levels: one level that relates to the single piece of software – to the atomic requirement – and another that relates to the whole integrated service environment.

Certainly the atomic aspects are already solved quite well and the assessment is as follows:

Drawback analysis:

- Likelihood: Low.
- Severity: Low.

Looking to the whole integrated aspect the risks seem to be a little higher:

Drawback analysis:

- Likelihood: Medium.
- Severity: Medium.

3.3.2 Service Issues

In a virtualised infrastructure, there will be (at least) two levels of control and management:

- Control and management of real physical infrastructure.
- Control and management of virtual infrastructure (maybe even recursive).

These two levels, including possible recursive levels, may or may not belong to the same administration entity. The multi-level aspect of control and management can impose a risk on the overall reliability and control/management performance of the network, and may influence each of the items below.

3.3.2.1 *Operational issues*

The operation of VNS will require:

- Additional manpower (probably not significant).
- Additional knowledge.

The provision of VNS requires additional effort for installation, configuration, maintenance, etc.

A major aspect is the increased complexity of the whole operational environment (e.g. additional service layers and network layers). An important requirement is that the provisioning operation related to the service VNS1 should not impact service VNS2. However, there might be side effects (interferences, interruptions) on other virtual networks of the same environment (e.g. performance degradation) and even on other network services in the same physical environment.

Even the (theoretical) isolation/separation of services cannot exclude such effects for sure. To minimise such side effects requires robust technical environments and operational processes. This will help to identify the issue and then where the responsibility for solving the problem lies.

In summary the operational requirements will certainly increase but with medium impact.

Drawback analysis:

- Likelihood: Very high.
- Severity: Medium.

3.3.2.2 *Security – general*

From a user point of view, virtualisation means the parallel but seamless use of services/components with other users. Users are in principle interested in having strict borders between themselves and other users; they like to have the impression of being the sole user of a dedicated service. On the other hand, among users in non-business environments there is a certain lack of concern about privacy/security aspects, as long as their service requirements (easy to use, inexpensive, always available) are fulfilled (e.g. the ongoing reports about privacy problems in Facebook, Google, etc. do not really worry them).

This is different, of course, in business environments (i.e. for companies), but also in big-science project environments within the research area (e.g. LHC). Maybe there is a useful distinction to be made between the awareness of security risks among companies/organisations/projects and individuals.

Security will be a major issue for the service provider, especially to avoid the introduction of back doors from the virtual environment to their network. However, it is probably not that important for the users in the NREN community (beyond the normally available security).

Another but related issue is to identify who should solve a security problem. Is the service provider also responsible for the inner aspects of a VNS that he gave to the user/researcher? What are the borders between both parties?

Drawback analysis from user point of view:

- Likelihood: Medium.
- Severity: Medium.

Drawback analysis from provider point of view:

- Likelihood: High.
- Severity: High.

3.3.2.3 *Security – user direct management*

Another aspect of security arises if in some virtual network environment users are allowed to define their own resources. This results in a certain kind of intervention in the virtual environment, which is related to security aspects. Here, the service provider has to define how much influence a user may have towards the definition of the virtual environment (and its alteration), i.e.

- What are the conditions, restrictions, limitations to accessing the virtual-service?
- What are the conditions, restrictions, limitations to accessing the virtual-environment, especially the management of virtual-components?

Drawback analysis from provider point of view:

- Likelihood: High.
- Severity: High.

3.3.2.4 *Failures, interruptions*

Which kinds of special problems exist within virtual environments? Are potential failures limited to the virtual environment of the users? Certainly not always, as total isolation will never be fully reached. Could failures within the VNS influence other services (outside of the VNS)? Again, sometimes probably yes.

Thus, there will be the open problem of what is the impact of problems on neighbouring virtual environments or other services? Even if actual failure is not caused, performance degradation could occur.

Especially if failures impact several services, the question arises of who is responsible for which failures and who has to start which actions to solve the problem?

A benefit of virtualisation is that the virtualised systems can take advantage of the host system's backup and recovery mechanisms. This means, however, that the host system's backup and recovery must be entirely robust. If there is a catastrophic problem, then the VNS may become a global single point of failure for many systems.

Drawback analysis:

- Likelihood: Low
- Severity: Tolerable

3.3.3 Business Issues

Even if the technical and service aspects outlined above could be solved, there might be other, business factors that restrict or hinder the introduction of VNS, such as those relating to the finding of an appropriate niche for a VNS, its cost, and independence from specific vendors (i.e. the extent to which an open system environment can be achieved).

This analysis is looking at users neither as part of commercial companies nor as private persons (users of commodity services) but as part of the R&E community (perhaps making a distinction within that group between big science projects such as LHC and individual interactions). This means it is not discussing the introduction of VNS in general but within the NREN community in particular, and the following aspects must be considered with regard to the specific goals and requirements of the NRENs.

3.3.3.1 *Business case*

A new service must offer added value for the user compared to existing services, and address a real demand. Thus it is important to be able to answer positively to such questions as:

- Does the service provide demonstrable added value to users?
- Does the service offer something new/different compared to existing services?
- Is the service one for which users are prepared to be charged extra costs?

It is also important to consider the positioning of the service relative to others and its potential impact on them, by answering questions such as:

- What are the competing services?
- Will there be cannibalism from other services in terms of both overlapping or appropriated functionality and users?

Ultimately, user requirements and user demand are the most important criteria for judging the feasibility of a VNS. To find the appropriate niche considering all the questions above could be a major problem.

Drawback analysis:

- Likelihood: High.
- Severity: High.

3.3.3.2 *No match with market demands*

As part of the general business case outlined above, the offered VNS could have implementation properties (e.g. service aspects, costs, complexity of use) that do not match market demands/expectations. This could hinder its introduction, even if in principle a niche has been identified, and make the investment worthless. A similar outcome could result from the appearance of other, more successful commercial solutions on the market.

Here one has to consider the special NREN community. A VNS should (and certainly will) be oriented towards the specific requirements of the NREN user, seen as a part of a scientific community (and not as a general private user). Thus the assumption is that an NREN VNS will not be similar to or in competition with the already available commercial services (e.g. from Google, Amazon, etc.) but will be directed to special needs and implemented with special properties.

Given the above assumption, and as, at the beginning, the investment will certainly be limited, the potential loss on investment will also be limited.

Drawback analysis:

- Likelihood: Medium.
- Severity: Tolerable.

3.3.3.3 *Costs*

A virtual service will generate some extra costs. The amount of such extra costs depends on the kind of service (e.g. which layer). An open issue will be the cost model towards the users; the two extremes are full costs or zero costs (i.e. hidden in other service offers). However, defining the appropriate cost model is outside the scope of this report and must be discussed elsewhere.

The cost elements include:

- Capital expenditure (hardware, software).
- Operational expenditure (including knowledge enhancement).

It is probable that operation and maintenance will not generate significant extra costs, provided the technology and the operational processes stay simple compared with non-virtual services. Otherwise the costs could easily increase considerably.

In summary, the cost issue will certainly arise but the additional costs should not be significantly high.

Drawback analysis from provider point of view:

- Likelihood: High.
- Severity: Medium.

3.3.3.4 *Inefficient use of resources*

A primary objective of virtualisation is to allow the efficient use of physical resources by abstracting the resources in such a way that they can be allocated on demand and returned when not needed. This can allow physical resources to take advantage of statistical multiplexing in a way similar to IP traffic's efficient use of large-capacity pipes.

However, if resources are allowed to be reserved – for example, if Layer 2 circuits are created with guaranteed bandwidth – then the opposite effect can take place: reserved resources might not be used, yet they are unavailable to other applications. Thus, in reality, certain operational restrictions will probably be implemented.

Inefficient use of resources also relates to aspects already discussed, e.g.:

- The quality of the service, which might suffer.
- The rising costs for the service provider.
- The cost model for the user, to prevent such inefficient use.

Drawback analysis:

- Likelihood: High
- Severity: Tolerable

3.3.3.5 *Delayed service introduction*

There seems to be no issues about the time of service introduction.

Drawback analysis:

- Likelihood: Low.
- Severity: Insignificant.

3.3.3.6 *Standards and maturity*

Currently, there are a number of vendor-specific commercial VNS available. At the same time, many people around the world are working towards generic solutions, independent of vendors, realising a kind of open system. However, such solutions are not yet mature enough for general and easy operation.

Users therefore currently have two options. They may either use a vendor-specific solution, which cannot readily be adapted to special needs and must be taken as it is. This not only leads to technical restrictions but also includes political aspects, e.g. becoming dependent on special vendors (e.g. security, even espionage). On the other hand, they could implement open solutions which are still under development and require a large amount of operational support.

Both options restrict the extensive, flexible and easy operation of VNS. That might change in the (possibly near) future with regard to an open solution. However, the current situation has some bearing on the questions mentioned in Section 3.3.3.1 *Business case*. In the meantime, the following assumptions have been made:

Drawback analysis:

- Likelihood: Medium.
- Severity: High.

3.3.3.7 *Organisational aspects*

Experience has shown that new technologies can influence organisational structures. In the NREN environment the introduction of VNS could, for example, change the role of computing centres, which could have a retarding influence to keep things as they are. However, that aspect remains very vague and is mentioned here mainly for completeness.

Drawback analysis:

- Likelihood: Low.
- Severity: Insignificant.

3.4 Summary Table

Table 3.1 below presents a summary of the drawback analysis.

Area	Aspect	Likelihood	Severity
Technical	Hardware environment	Low	Tolerable

Area	Aspect	Likelihood	Severity
	Software environment: • Atomic requirement • Whole integrated service environment	• Low • Medium	• Low • Medium
Service	Operational	Very high	Medium
	Security – general: • User point of view • Provider point of view	• Medium • High	• Medium • High
	Security – user direct management: • Provider point of view	• High	• High
	Failures, interruptions	Low	Tolerable
Business	Business case	High	High
	No match with market demands	Medium	Tolerable
	Costs: • Provider point of view	• High	• Medium
	Inefficient use of resources	High	Tolerable
	Delayed service introduction	Low	Insignificant
	Standards and maturity	Medium	High
	Organisational aspects	Low	Insignificant

Table 3.1: Drawback analysis summary

3.5 Discussion and Conclusions

This drawback analysis has considered areas of possible problems that could hinder the introduction of Virtual Network Services as a major service component of the NREN service portfolio. The orientation towards the NRENs takes into account their special role and situation. Several of the problems discussed above are also true for other service providers, but some are especially valid for NRENs, such as user demand or service cannibalism in relation to their existing services.

Specifically, no major issues have been identified with regard to technical features; the hardware is able to provide the necessary capabilities and the general software is also able to provide such functionalities.

However, apart from these purely technical aspects, somewhat larger problems still exist, especially with regard to the operational environment, the maturity of solutions and the area of security. These items are less important when dealing with project-internal or otherwise limited service requests, but they become very important for full service provision.

A further area of uncertainty relates to real user requirements and possible service cannibalism seen from the point of view of NRENs. Users are not interested in Virtual Network Services for their own sake; they are interested in network services that fulfil their demands. Such demands, with their requirements for functionality, flexibility and cost, might be provided via traditional services or more easily via virtual networks. However, the preferred solution will be evaluated by the service provider (in this case, NREN) not by the users (except in so far as the users get the service they require).

There are a number of further aspects (e.g. efficiency, troubleshooting, organisational items) that are less important. Such items must be improved but they play no central role for or against the introduction of a Virtual Network Service.

As a conclusion, there are no insuperable obstacles to the introduction of virtual networks. However, there are a number of small, medium and even sometimes large drawbacks to its full introduction. Thus it will always be a matter of evaluating the advantages and added value compared with traditional services, and of assessing the impact of the gaps that still exist in some areas (e.g. operation, security) when considering the operation of Virtual Network Services currently or for the foreseeable future.

4 GÉANT Virtualisation Service (GENUS)

4.1 Introduction

Virtualisation, in general, is by now a common activity. Operating platforms, software, storage and/or processing resources are being provided as virtualised services (by Amazon and Google, for example). Generally, the layers seen in the market are Application, Platform and Infrastructure, which define three generic and well-known business models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), respectively. Cloud computing is related to the above business models.

Network virtualisation provides the basis for the so-called Network as a Service (NaaS). The concept behind NaaS is analogous to the widespread IaaS: provide the consumer with simple but powerful tools to use and operate the infrastructure resources, together with attractive rates and payment models (e.g. pay as you go). Unfortunately, NaaS is lagging behind, mainly due to the lack of flexible and automated commercial services such as lambdas, Ethernet and IP services. It is in order to alleviate this fact that JRA1 Task 4 is creating a mechanism for federated virtualised networks called GÉaNt virtUalisation Service (GENUS).

This chapter defines a virtualisation service within the context of GÉANT and proposes an approach to its implementation within GÉANT and associated NREN infrastructures. The recommendations in this chapter are based on the results of the comprehensive study of existing virtualisation technologies reported in Chapter 2 and the initial requirements analysis reported in [GN3-DJ1.4.1] Section 3.

JRA1 Task 4's proposal for a GÉANT virtualisation service aims to take advantage of each of the existing relevant European projects and initiatives (the Task participants' involvement in these projects means they are well placed to leverage the first-hand knowledge and experience gained), while providing the capability to incorporate the outcome of any future relevant projects and frameworks. JRA1 Task 4 is not aiming to promote a specific solution or framework for the GÉANT virtualisation service. Instead, it aims to propose a solution for integrating and interworking existing virtualisation mechanisms and solutions at different layers, leaving the choice of suitable virtualisation technologies for each domain to individual NRENs.

JRA1 Task 4 has defined the required virtualisation services within GÉANT and associated NRENs in four different layers, as described below:

- Computing virtualisation: aggregating several computing servers or partitioning a server into several independent servers by means of an operating system.

- Layer 3 network virtualisation: creating Layer 3 (IP)-related functionalities on any type of hardware. This includes partitioning a Layer 3 router into several independent routers to create a Layer 3 virtual network topology.
- Layer 2 network virtualisation: creating Layer 2 (Ethernet)-related functionalities on any type of hardware. This includes partitioning a Layer 2 switch into several independent switches to create a Layer 2 virtual network topology.
- Layer 1 (optical) network virtualisation: creating a Layer 1 network topology by binding together Layer 1 resources (e.g., SDH timeslots, wavelength, fibre). This includes partitioning (slicing) of Layer 1 devices such as optical switches.

In each of the above layers, virtualisation can occur according to the user's community needs. Indeed, projects such as LHC, for instance, could request from an NREN and/or GÉANT a dedicated Layer 3 VPN. The French Grid Research Infrastructure GRID5K has its own physical optical VPN on top of RENATER infrastructure. The JIVE projects EXPReS and NEXPReS rely on a set of stitched lightpaths that is also called a set or string of Single Point of Failure.

As described in Chapter 2, there are various initiatives and projects focusing on virtualisation services and technologies. However, each of these projects is focused on a specific area and their solutions only deal with a restricted number of layers. Without reinventing the wheel, JRA1 Task 4's proposal is to integrate existing Layer 1, Layer 2, Layer 3 and computing virtualisation tools. Based on this aim, this section defines the GÉaNt virtUalisation Service (GENUS) and an architecture for it. GENUS architecture is a multi-layer, multi-domain and multi-technology virtualisation architecture suitable for NREN and GÉANT requirements, leveraging tools and software that have already been developed or are currently under development within GÉANT and the European research community.

GENUS is based on the following fundamental assumptions:

- GENUS is not a virtualisation mechanism or framework. It leverages virtualisation frameworks, mechanisms, tools and software already implemented within various EU projects and initiatives as well as the GÉANT bandwidth-on-demand provisioning tool (AutoBAHN).
- GENUS requires NRENs to adopt an existing virtualisation mechanism. The choice of virtualisation framework and mechanism is up to each NREN based on their requirements and constraints.
- NRENs and the GÉANT backbone network are the infrastructure providers for GENUS; GENUS itself has no resources.

As in the drawback analysis, the term "user" refers broadly to any GÉANT and NREN users, the target audience for the GENUS service, who need their own infrastructure and control over it. An initial view of what NREN users might use GENUS for was obtained from the requirements survey, documented in [GN3-DJ1.4.1] Chapter 3.

Note that a consideration of cost and associated aspects such as a cost process and model is outside the scope of the Task.

4.2 GENUS Services

The main service that GENUS aims to offer is on-demand provisioning of end-to-end multi-domain multi-layer virtual infrastructure (network infrastructure + IT infrastructure) over the GÉANT community, leveraging the capabilities of NRENs' virtualisation mechanisms as well as GÉANT's bandwidth-on-demand provisioning tool. To achieve this objective, GENUS needs to provide an abstraction layer for the virtualisation mechanisms adopted by the NRENs, hiding from the users the complex technical details and the heterogeneity of the different frameworks. GENUS will have to provide a set of basic functionalities and tools as described below and depending on the actors that will interact with it:

- Registration and advertising.
  - NRENs supporting a virtualisation mechanism, either delivering network resources (such as virtual circuits, virtual routers and switches) or raw computing elements as virtual machines, will be able to register for the service.
  - Once registered they will be able to advertise their virtualisation framework through a specific interface, specifying to the service the resources that will be available for leasing to the users.
- End-user interface.

  The end users will interact with GENUS using a graphical user interface (GUI). Through this abstraction they will be able to access the underlying virtualisation platform transparently. In particular, they will be able to access features such as:

  - Discovery of resources. The users will be able to access a list of the resources that NRENs expose to GENUS. The system will expose information such as the nature of the resources that can be included in a slice (network and IT) and their location. In addition, GENUS will provide information about the capabilities and services of the available virtual resources. This way the user will be able to filter the resources and select the ones that best fit the needs of the slices.
  - Virtual resources allocation and virtual infrastructure composition. GENUS will accept the abstracted description of a slice (virtual infrastructure) and will convert it into a complete slice, interacting with the NRENs' virtualisation frameworks. The technical details of the whole allocation process are transparent to the final users. Once the process is complete, GENUS returns the users an end point (e.g., a URL) from where they will be able to interact with their slices.
  - Operation, control, management and monitoring of the virtual resources and infrastructure. GENUS will provide a dashboard from where users can monitor the activities of the slices' elements. GENUS will also offer a mechanism for accessing the individual resources, so that the users can control and configure the behaviour of slices' elementary blocks. (This feature has to be supported by the NRENs' virtualisation mechanism. FEDERICA does not support it, but most other mechanisms do.)
  - Release of resources. Finally, GENUS will provide an interface to let the user release their slices and advertise the resources as available for creating new virtual infrastructures.

Figure 4.1 below shows a use-case diagram of the GENUS system with the basic functionalities described above.

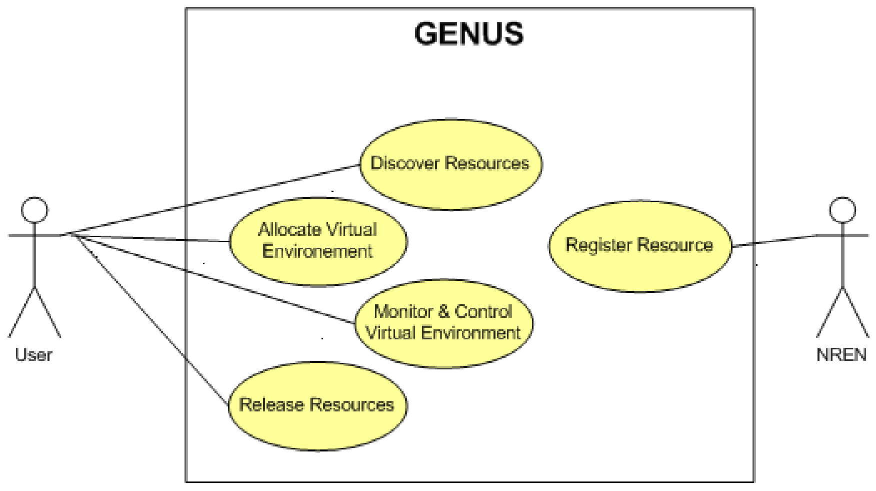

GENUS
Discover Resources
Allocate Virtual Environement
Register Resource
Monitor & Control Virtual Environment
Release Resources
User
NREN

Figure 4.1: GENUS basic functionalities with its main actors

GENUS leverages NRENs' virtualisation mechanisms and GÉANT's bandwidth-on-demand (BoD) provisioning service (GENUS interfaces with the AutoBAHN tool for BoD), enabling the composition and operation of federated multi-layer virtual infrastructure. The NREN's virtualisation mechanism is responsible for creating the virtual infrastructure (infrastructure slice) within each NREN infrastructure, while GENUS performs orchestration and federation. In other words, GENUS is an orchestration and stitching mechanism, which is able to compose a federated virtual infrastructure made of two or more slices of NRENs' infrastructure, created by the NREN's virtualisation mechanism and interconnected by GÉANT infrastructure using its GÉANT BoD service. GENUS is also able to abstract and hide all technological details and interfacing complexity from users and provide a mechanism where users can communicate with all resources in a uniform and abstract way.

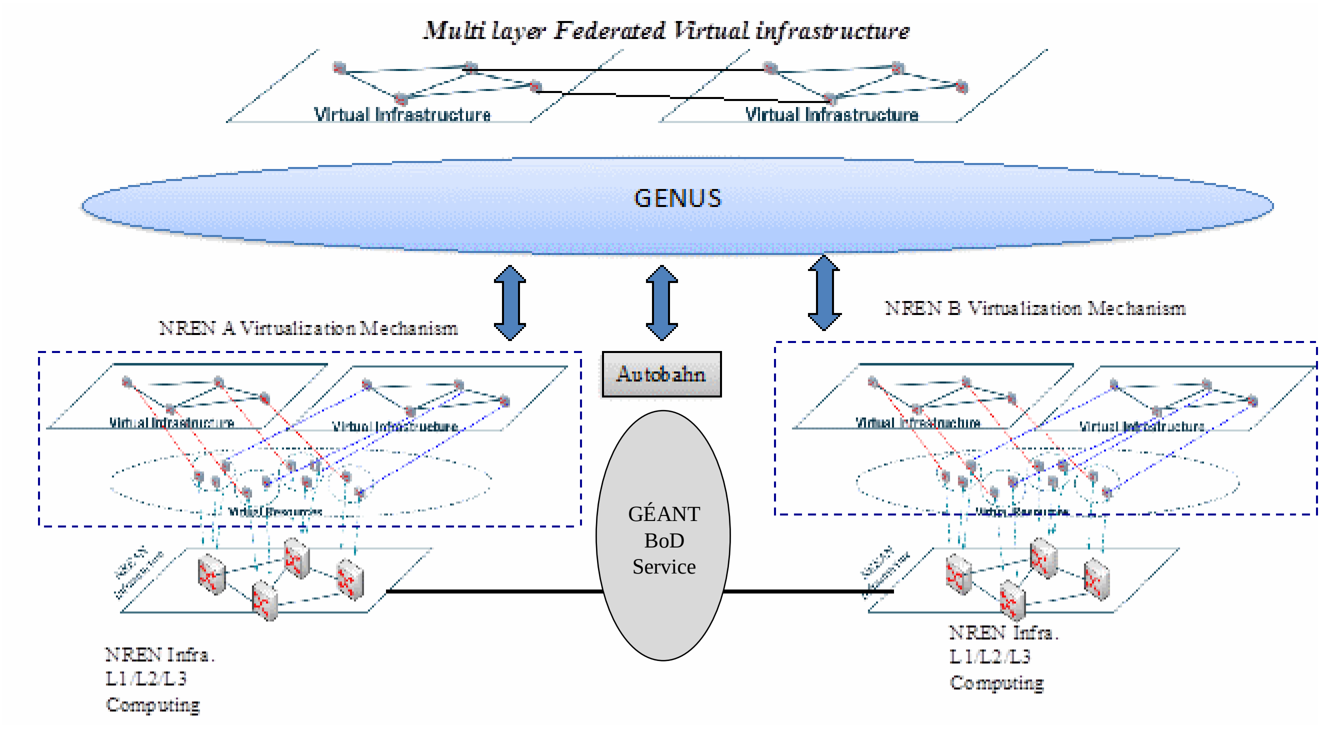

Multi layer Federated Virtual infrastructure
Virtual Infrastructure
Virtual Infrastructure
GENUS
NREN A Virtualization Mechanism
NREN B Virtualization Mechanism
Autobahn
GÉANT BoD Service
Virtual Infrastructure
Virtual Resources
NREN Infra. L1/L2/L3 Computing
NREN Infra. L1/L2/L3 Computing

Figure 4.2: GENUS's role in providing federated multi-domain, multi-technology virtualisation

4.3 Generic GENUS scenario/use case description

For a better understanding of the expected behaviour of GENUS, this section gives a step-by-step description of what a user will do to instantiate a virtual infrastructure. For simplicity, the scenario will consider a minimal set of resources and functionality that will be implemented in the first GENUS prototype. (A more detailed description of the prototype functionalities and the demonstration testbed infrastructure is provided in Chapters 6 and 7.)

Step	Action	Description
1	User accesses GENUS.	The user logs in to the GENUS GUI.
2	User selects resources.	The user accesses the repository of resources registered by the NRENs as being available for including in virtual infrastructures (slices). The GUI provides information about the characteristics and capabilities of the virtualisation platforms. With this information the user can query the repository for an NREN providing the desired features: for example, L2 slicing/virtualisation with a given bandwidth and virtual machines with given computing capabilities (number of virtual cores, RAM and disk size, number of virtual NICs). In addition, the user will be able to choose the end point that will be used to access the virtual infrastructure data plane. In the prototype, geo-location of the nodes will be used to indicate the location of the resources.

Step	Action	Description
3	User composes virtual infrastructure (slice) description.	By choosing the network and computing elements from the pool of resources obtained in the previous step, the user composes a global description of the virtual infrastructure (slice). The description includes the location of the end nodes, the topology and the capabilities of the single virtual resources.
4	User submits the virtual infrastructure (slice) description to GENUS.	Once the user has submitted the slice description, GENUS breaks down the requirements for individual NRENs and translates them to their associated provisioning system APIs (i.e. the APIs of the NREN's virtualisation system). GENUS then informs the user about the success of the instantiation process. If it is successful, GENUS provides a unique ID, or a URL, from where the user can control and monitor the status of the environment. In case of failure, GENUS logs the reasons that prevented the creation of the slice and reports them to the user for further investigation.
5	User controls, manages and monitors the slice.	Using the dashboard that will be provided in the final version of GENUS, the user manages, controls and monitors the behaviour of the resources in a running slice. In the first version of GENUS, monitoring features will be minimal and will provide a health check on the resources. The control and management features will be provided to users as URLs for the specific control systems of the individual virtualisation frameworks in each NREN.

Table 4.1: Steps to instantiate a virtual infrastructure

4.4 State-of-the-art virtual infrastructure federation

One of the main objectives of GENUS is to provide a virtual infrastructure made of infrastructure slices from multiple NRENs interconnected by GÉANT. To achieve this, GENUS aims to provide a mechanism for federation and orchestration of infrastructure slices from different NRENs. From the control and management plane's point of view, there are two different approaches to federating infrastructures. First, there is the top-down federation approach adopted by Teagle in the European Panlab testbed [Panlab]. Second, there is the bottom-up federation approach, represented by Slice-based Federation Architecture (SFA), widely adopted in the US (GENI) and in Europe (OneLab) [SFAv1, SFAv2, SFAOver]. Each of these is described below. (The information is based on the NOVI deliverable "D3.1 State-of-the-Art Management Planes" [NOVI-D3.1].)

4.4.1 Teagle

The Pan-European Laboratory for Next Generation Networks and Services, Panlab, and its successor, the project Panlab Infrastructure Implementation, PII [Panlab], address the need for large-scale testing facilities in the communications area by implementing an infrastructure for federating testbeds. To facilitate the technical aspects of testbed federation, Panlab relies on the Teagle framework, a web instance that provides the means for a Panlab customer to specify his testing needs and get feedback on where, how and when testing can take place.

The Teagle Tool can be accessed by Panlab partners and customers via the Teagle Portal, a simple web interface where the testbed partners can enter the relevant data describing their testbed and its resources, and Panlab customers may then search the Panlab Repository to find resources suitable for their tests, or they can specify their testing requirements and get feedback on where, how and when testing can take place. Panlab users are authenticated on the common web interface of the Teagle Portal. All the information required for internal authentication and authorisation in Panlab is stored in the Teagle Repository, secured by HTTP Authentication.

From the Teagle Portal, customers can launch the VCT Tool to create Virtual Customer Testbeds (VCTs). The VCT Tool is a Java Web Start application that communicates at start-up and during runtime with the Teagle Repository. The Teagle Repository stores the common information model for high-level agreement across the tool set and domain-specific data models for separation of concerns. The communication with the repository is done via the same Java classes as the VCT tool uses for repository queries.

The Teagle Tool also includes an Orchestration Engine (OE) component, which allows users to specify requests to the federated testing environment. These requests are mapped against existing services exposed by the testbed. In general, more than one testbed is used in a federated environment, which requires a method for combining and synchronising various unconnected components. The testbed orchestration system provides such collaborative processes, starting from definition of the user request to the actual execution of an orchestration script.

The Panlab Testbed Managers (PTM) are installed at every testbed whose resources are intended to be offered through the Panlab platform. PTMs implement the interactions at the control layer between the setup and configuration requests by Teagle and the components in the testbed it manages. It translates generic Create, Read, Update, Delete (CRUD) commands, received from Teagle as Representational State Transfer (REST) messages, into resource-specific communications (e.g. SNMP), that is, control commands applicable to the testbed component.

A PTM consists of two parts: the Core PTM and the Resource Adaptation Layer (RAL). The role of the RAL is to integrate the resources offered by testbed owners to the PII platform. The resources are controlled by resource-specific Resource Adapters (RAs). They provide a common interface for communication with PTM modules while implementing resource-specific communication and configuration protocols when dealing with resources. They act like device drivers for the different testbed resources. The RA common interface communication schema follows a generic XML schema catering for the description of all the possible configuration parameters a resource may be publishing.

Teagle's architecture provides a hierarchical federation model that is shown in Figure 4.3 below.

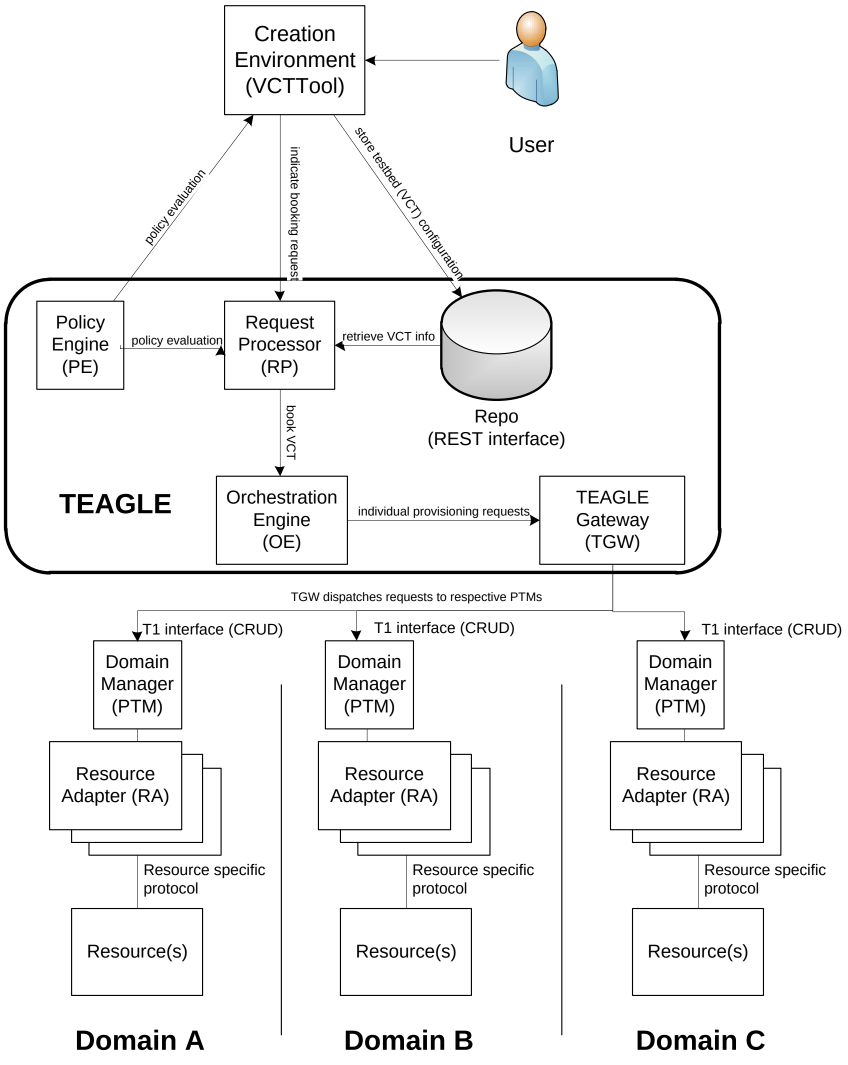


Figure 4.3: Teagle overview

4.4.2 Slice-based Federation Architecture (SFA)

The Slice-based Federation Architecture (SFA), as described by the SFA Draft 1.0 [SFA1], then by the SFA Draft 2.0 [SFA2], and implemented by PlanetLab [PlanetLab, SFAimpl] is an API specification and a software system that allows different testbeds to federate, i.e. a user registered on testbed X can access resources in testbed Y and Z too in a transparent way. The description of SFA provided in this section borrows largely from the SFA Draft, but also includes the pragmatics developed via the PlanetLab implementation of the SFA system and its daily use.

SFA defines the minimal set of interfaces and data types that enable a federation of slice-based network components to interoperate. SFA defines two key abstractions: components and slices.

- Components are the primary building block of the architecture. A component might correspond to an edge computer, a customisable router, or a programmable access point. A component is comprised of a collection of resources, including physical resources (e.g., CPU, memory, disk, bandwidth) logical resources (e.g., file descriptors, port numbers), and synthetic resources (e.g., packet-forwarding fast paths). These resources can be contained in a single physical device or distributed across a set of devices, depending on the nature of the component. Each component is controlled via a component manager (CM), which exports a well-defined, remotely accessible interface. (Components can be grouped into aggregates, with each aggregate controlled by an aggregate manager (AM) that plays the same role as a component manager). The component/aggregate manager defines the operations available to user-level services to manage the allocation of component resources to different users and their experiments.
- A slice is defined by a set of resources spanning a set of network components, plus an associated set of users that are allowed to access those resources for the purpose of running an experiment on the substrate. The slice manager (SM) is a proxy between user and aggregate managers (AMs), which represent a collection of components as a single aggregate for slice operations, and decides which AM to contact. The registry (R) maintains information about a hierarchy of management authorities and maintains information about a hierarchy of slice authorities.

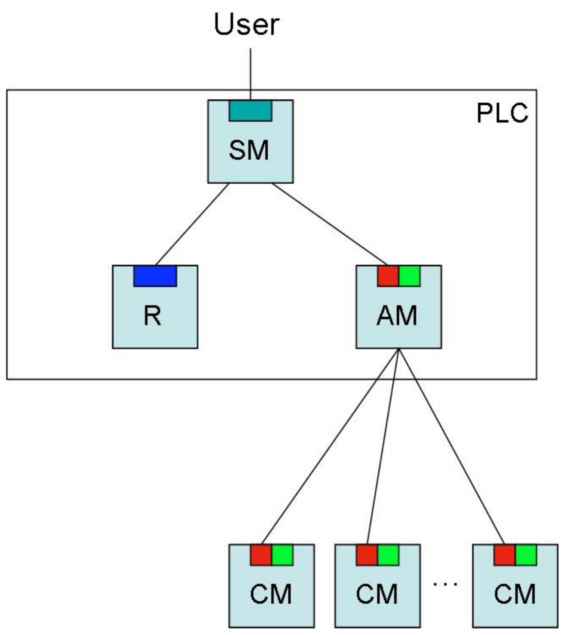


Figure 4.4: PlanetLab SFA architecture

SFA gives users access to heterogeneous resource types. The resource specification (RSpec) is the means that SFA uses for declaring those resources. RSpecs provide a language for describing the resources (both physical and logical) exported by an aggregate (collection of resources). So far, SFA has taken a bottom-up approach to defining the RSpec, allowing each new type of aggregate to specify its own RSpec format using XML. The RSpec serves two purposes: to let the aggregate advertise information to the user about the available resources, and to enable the user to request a subset of the resources to be allocated to a slice. The aggregate manager is responsible for generating and processing RSpecs. Implementing a new RSpec requires changes in the aggregate manager.

By formalising the interface around the slice, resource owners and users are free to cooperate more easily. Owners simplify the administrative overhead of making their systems easily accessible to more users, and users gain access to interesting systems without the overhead of setup and administration.

4.5 GENUS architectural building blocks

GENUS's architecture matches the Teagle approach better than the SFA approach, because the context is of testbeds that already exist and the objective is to set up a layer on top of them in order to federate them. The GENUS federation concept is closer to a hierarchical federation than a peer-to-peer federation. The architecture that is proposed in this section is therefore more inspired by Teagle's architecture than by SFA. However, the adoption of the Teagle model would not preclude the possibility of adopting some benefits provided by the SFA approach. As already mentioned, the main service that GENUS aims to offer is on-demand provisioning of end-to-end multi-domain multi-layer virtual infrastructure (network infrastructure + IT infrastructure) over the GÉANT community, leveraging the capabilities of NREN's virtualisation mechanisms as well as GÉANT's bandwidth-on-demand provisioning tool.

The proposed GENUS architecture is depicted in Figure 4.5 below.

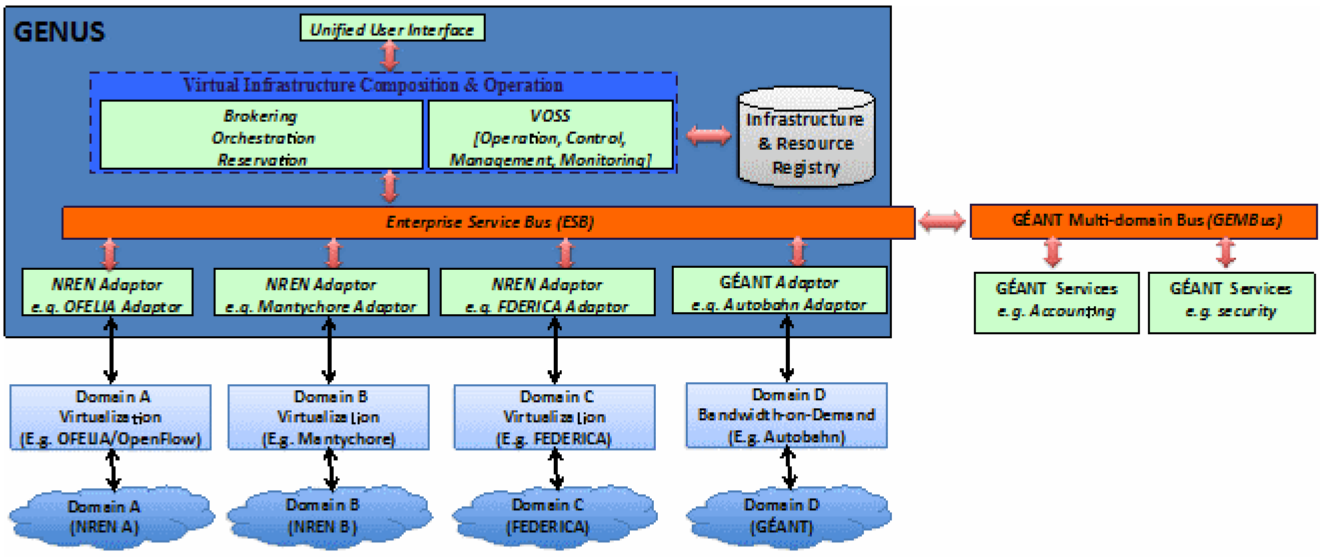

GENUS
Unified User Interface
Virtual Infrastructure Composition & Operation
Brokering Orchestration Reservation
VOSS [Operation, Control, Management, Monitoring]
Infrastructure & Resource Registry
Enterprise Service Bus (ESB)
GÉANT Multi-domain Bus (GEMBus)
GÉANT Services e.g. Accounting
GÉANT Services e.g. security
NREN Adaptor e.g. OFELIA Adaptor
NREN Adaptor e.g. Mantychore Adaptor
NREN Adaptor e.g. FEDERICA Adaptor
GÉANT Adaptor e.g. Autobahn Adaptor
Domain A Virtualization (E.g. OFELIA/OpenFlow)
Domain B Virtualization (E.g. Mantychore)
Domain C Virtualization (E.g. FEDERICA)
Domain D Bandwidth-on-Demand (E.g. Autobahn)
Domain A (NREN A)
Domain B (NREN B)
Domain C (FEDERICA)
Domain D (GÉANT)

Figure 4.5: GENUS architecture

The blocks that make up the GENUS architecture are as follows:

- **Unified User Interface:** a web service for users where they can communicate with the GENUS system. Using this interface, the users can obtain a list of participating NRENs and their available resources and capabilities. They can also submit their request for the allocation of the resources they specify.

- **NREN Adaptor:** an interface, adaptor and translator to the NREN virtualisation mechanism. It connects the GENUS system to the NRENs' virtualisation systems via a set of APIs available from the NRENs' virtualisation mechanisms (i.e. it translates users' requirements into a format compatible with the APIs of the NREN-specific virtualisation mechanism).

- **GÉANT Adaptor:** translates users' requirements into a format understandable by AutoBAHN. In the GENUS system, AutoBAHN is used for provisioning inter- and intra-virtual infrastructure connectivity.

- **Resource Registry:** a mechanism that provides the capability for NRENs to register and modify their infrastructure and virtualisation mechanisms in the GENUS system for user access.

- **Virtual Infrastructure Composition (Brokering, Orchestration, Reservation):** a set of functionalities that breaks down the user's request, sends the appropriate request(s) to one or multiple NRENs, reserves resources and creates the requested virtual infrastructure.

- **Virtual Infrastructure Operation:** a set of functionalities that allows a user/owner of a virtual infrastructure to control, monitor, configure and manage virtual resources. This is also called the Virtualised Operations Support Service (VOSS), and is explained in more detail in Chapter 5.

- **Enterprise Service Bus (ESB).** GENUS will adopt an architecture based on the Service-Oriented Architecture (SOA) model. In particular, it will exploit the functionalities provided by an Enterprise Service Bus middleware. Under this assumption, the distinction between a centralised and a distributed implementation becomes blurred: the GENUS functionalities are modules of the ESB [Fuse] and their deployment pattern affects the nature of the whole service. If the components are hosted by a single container (i.e. the element of the ESB that runs the architecture modules), then GENUS will behave like a centralised system, interacting with the clients through a unique end point (the GUI). By contrast, if more ESB instances are used for the deployment then different interaction patterns will let GENUS appear as a cloud.

  Having a distributed layout of the ESB platforms does not affect the behaviour of the GENUS components. It introduces additional features for the resiliency and scalability of the service. In particular, high availability and load balancing can be configured so that the system can scale horizontally to manage the workload dynamically.

  It is relevant to observe that the presence of an ESB makes the deployment choice, i.e. distributed versus centralised, transparent for the clients. In the case of distributed layout, the only additional information of which the GUI is aware is that GENUS can be accessed from different end points.

  The distribution of the layout can be done at different levels: introducing high availability with static failover for the single components of the system, or creating a cluster of different ESBs sharing the containers and the message brokers. In the former configuration, replicas of the functional components of the systems are instantiated in the same container. Every instance has different end points and the client configuration file reports their existence. The client side remains unaltered; only the configuration of how the service is contacted differs. The replicas ensure that if a module of the system becomes unresponsive, the functionality is still available through cloned modules. ESBs like Fuse and GÉANT Multi-domain Bus (GEMBus) [GEMBus] offer different policies for accessing the replicas: round robin and random selection of the entry point. Random selection, in addition to high availability, provides a simple strategy for balancing the load among the instances of a module. Distribution can be done also by clustering together ESB servers.

The benefit of having clustered containers is greater than static failover for the single components: in addition to transparent load balancing and high availability (containers are aware of their peers for failover), rollback and redelivery of failed message exchanges is provided. Different communication patterns among the message brokers can be arranged to implement different distribution patterns: one-way forwarding for master-slave configurations or bi-directional connections for creating peer-to-peer islands. Figure 4.5 shows how different ESBs could be configured to deploy GENUS in a distributed fashion.

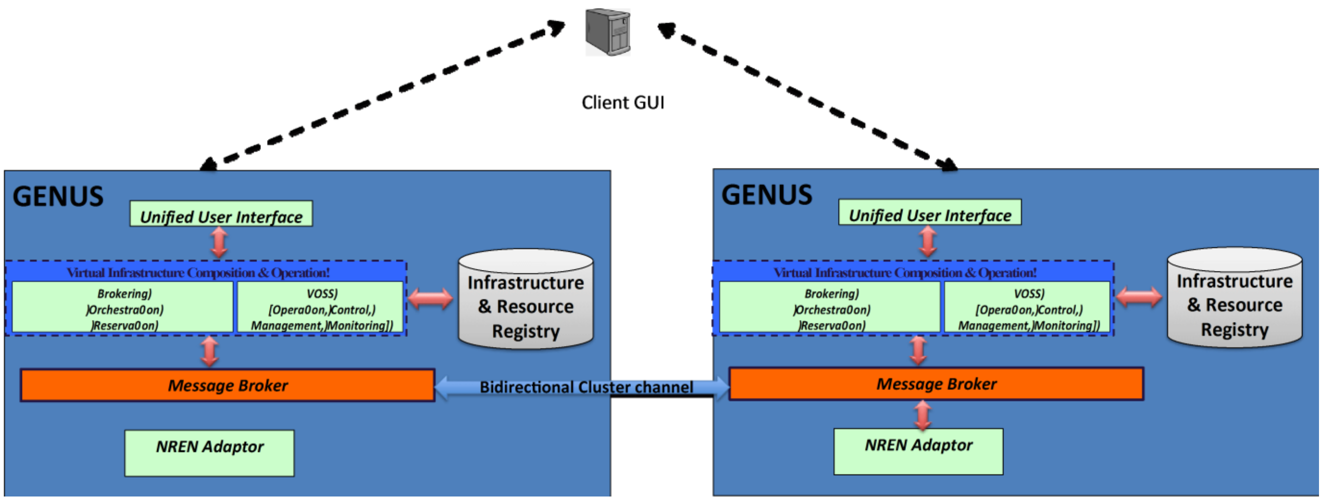

Client GUI
GENUS
Unified User Interface
Virtual Infrastructure Composition & Operation
Brokering))Orchestra0on))Reserva0on)
VOSS) [Opera0on,)Control,) Management,)Monitoring])
Infrastructure & Resource Registry
Message Broker
Bidirectional Cluster channel
NREN Adaptor
GENUS
Unified User Interface
Virtual Infrastructure Composition & Operation
Brokering))Orchestra0on))Reserva0on)
VOSS) [Opera0on,)Control,) Management,)Monitoring])
Infrastructure & Resource Registry
Message Broker
NREN Adaptor

Figure 4.6: Distributed GENUS layout using ESBs

If a component fails then the message broker of the master ESB transparently reroutes the requests to the corresponding replicated component hosted on the slave server. If the component connecting the ESB to the clients crashes, the static failover configuration deployed on the client ensures that users will continue to interact with GENUS by connecting to the interface of the slave ESB.

- **GEMBus Interface.** GENUS also deploys an interface to GÉANT Multi-domain Bus (GEMBus). It allows some additional functionalities and GÉANT services that already interface with GEMBus to be integrated with GENUS. Examples of these services are:
  - The GEMBus Accounting Service that will allow users to track their GENUS transactions.
  - The GEMBus Security Token Service (STS) that will provide secure GENUS transactions.
  - Access to AutoBAHN through GEMBus (in addition to direct interfacing via the AutoBAHN adaptor), since AutoBAHN is integrated with GEMBus.

It must be noted the building blocks mentioned above are the basic mechanisms that are required for the first version of GENUS. An operational GENUS service will require other services such as accounting, security, etc.

The functional building blocks shown in the GENUS architecture (Figure 4.5) are not restrictive with respect to the actual deployment of the GENUS components. According to the architectural models adopted by similar pre-existing middleware, some blocks can be located either on the client side (i.e., on the machine of the user requesting a virtual resource) or in a centralised service (leaving the client's layer as thin as possible). Other blocks, like the resource registry, collecting the resources exposed by the NRENs, have to remain functionally separated entities, because they store and manipulate shared information.

The actual GENUS architecture could be in any intermediate position between the opposite ends of the centralised/distributed spectrum. By following a lean software development philosophy, the final choice should be left to the further investigation of possible deployment use cases and applications, as well as trial results.

4.6 References

[Fuse]	Fuse ESB http://fusesource.com/products/enterprise-servicemix/
[GEMBus]	Diego R. Lopez, "The GEMBus Way: Delivering the Promise of the Internet of Services" http://www.terena.org/activities/eurocamp/nov09/slides/20091118-GEMBus-EuroCAMP-Budapest.ppt
[NOVI-D3.1]	NOVI deliverable "D3.1 State-of-the-Art Management Planes" http://www.fp7-novi.eu/index.php/deliverables/doc_download/24-d31
[Panlab]	http://www.panlab.net
[PlanetLab]	http://www.planet-lab.org
[SFAImpl]	PlanetLab Implementation of the SFA https://svn.planet-lab.org/browser/sfa/trunk/docs/sfa-impl-2009-04-07.pdf
[SFAOver]	SFA Overview http://svn.planet-lab.org/wiki/SFAGuide#SFAOverview
[SFAv1]	"Slice-Based Facility Architecture", Draft Version 1.04, April 7, 2009 https://svn.planet-lab.org/browser/sfa/trunk/docs/sfa.pdf
[SFAv2]	"Slice-Based Federation Architecture", Version 2.0, July 2010 http://groups.geni.net/geni/wiki/SliceFedArch

5 Virtualised Operations Support Service (VOSS)

5.1 Introduction

As described in Chapter 4, GENUS will create a federated multi-layer virtualised environment. The virtualised network service provided by the federated domain will be based on technologies that have been developed elsewhere, such as by European projects including GN3's AutoBAHN, MANTYCHORE, NOVI, Panlab and OFELIA, and by NRENs, including HEAnet (Bluenet) and SURFnet (OpenDRAC). This chapter describes GENUS' approach to virtualising the operational management functions of a resource through the Virtualised Operations Support Service (VOSS), a set of functionalities within GENUS that allows a user/owner of a virtual infrastructure to control, monitor, configure and manage virtual resources.

Most present-day technologies look at virtualising the service that is provided (such as a data plane being an IP network, an Ethernet connection, etc.). In VOSS, the Resource that is actually being used is called Worker Resource. Such a Resource can also be seen in other services based on IT services, for instance, storage or computer cycles.

Besides the Worker Resource, VOSS also defines the Management Resource, following the analogy of the management plane in telecommunications. The Management Resource virtualises the operational management functions of the Worker Resource. Utilising GN3's Network Management Architecture (NMA), which is based on TeleManagement Forum's (TM Forum's) Enhanced Telecom Operations Map (eTOM), the following Management Resources can be recognised:

- Configuration and Activation Resource (installing, configuring and optimising Worker Resource).
- Quality Management Resource (managing, tracking, monitoring and reporting on the performance of a Worker Resource).
- Trouble Management Resource (recognising, isolating and correcting faults).
- Policy Management Resource (allowing the addition, modification or deletion of policy rules and attributes with regard to the business logic).
- Information Management Resource (e.g. storing information related to Worker Resources).

VOSS combines the ecosystem of the Resource, Ownership, Role and Actors (RORA) model from GEYSERS (an EU FP7 project addressing infrastructure virtualisation for cloud services, described in Section 2.9) and a marketplace with the operational management of resources from MANTYCHORE (an EU FP7 project addressing IP infrastructure virtualisation, described in Section 2.3). The advantage of seeing Management Resources in the same way as Worker Resources is that it opens up the possibility to sell/subcontract, that is, trade, any (Worker and Management) Resources in the marketplace using normalised interactions between actors. VOSS aims to provide the required operational and support services for the GENUS system to pave the way to making GENUS an operational virtualisation provisioning service for NRENs.

As mentioned above, VOSS is based on the definition of Management Resources and Worker Resources. The relationship between both is many to one, but the proposed design allows for independent resource manipulation. Virtualisation processes that rely on Worker Resource aggregation are not very common. Many vendors offer solutions to manage networks (if Web Services-based: Worker Resources) using proprietary, centralised management systems (normally not Web Services-based Management Resource). These management functions are monolithically implemented in the management system. In the VOSS model, the function sets can be split and a Management Resource defined for each. This way, operators' management workflows are more flexible and allow them to incorporate or delegate actions to third parties, under certain conditions.

The Operational Management functions are related to the Operations Support Systems (OSS) mentioned in GN3's NMA (for more information, please refer to GN3 deliverable "DJ2.1.1: Information Schemas and Workflows for Multi-Domain Control and Management Functions" [GN3-DJ2.1.1]). The OSS can be based on TM Forum's Next Generation Operations Support System (NGOSS) Distributed Interface Oriented Architecture (DIOA). This architecture allows a resource-oriented style, which is aligned with the NaaS environment. Like the NaaS environment, the NMA uses a Service-Oriented Architecture (SOA), thus enabling a Web Service (WS) environment.

Furthermore the multi-administrative domain aspect (which is not part of eTOM) is included in GENUS/VOSS's vision.

It is important to mention that for virtualisation, and the possible marketplace that will evolve around it, multi-domain is more related to the physical domains: a user acquires and then owns the right to specific Resource parameters (under some "lease" restrictions), so this is slightly different to owning the Resource (this is comparable to renting a house). A user gathering such Resources becomes a single administrative domain (even though the virtual/physical Resources are provided by others). This layering and the possibility to re-market an (enhanced) Resource is important in a virtualised world. This recursive granting of rights is one of the key enablers of NaaS flexibility.

This chapter looks at the parts of eTOM Level 2 shown in Figure 5.1 below.

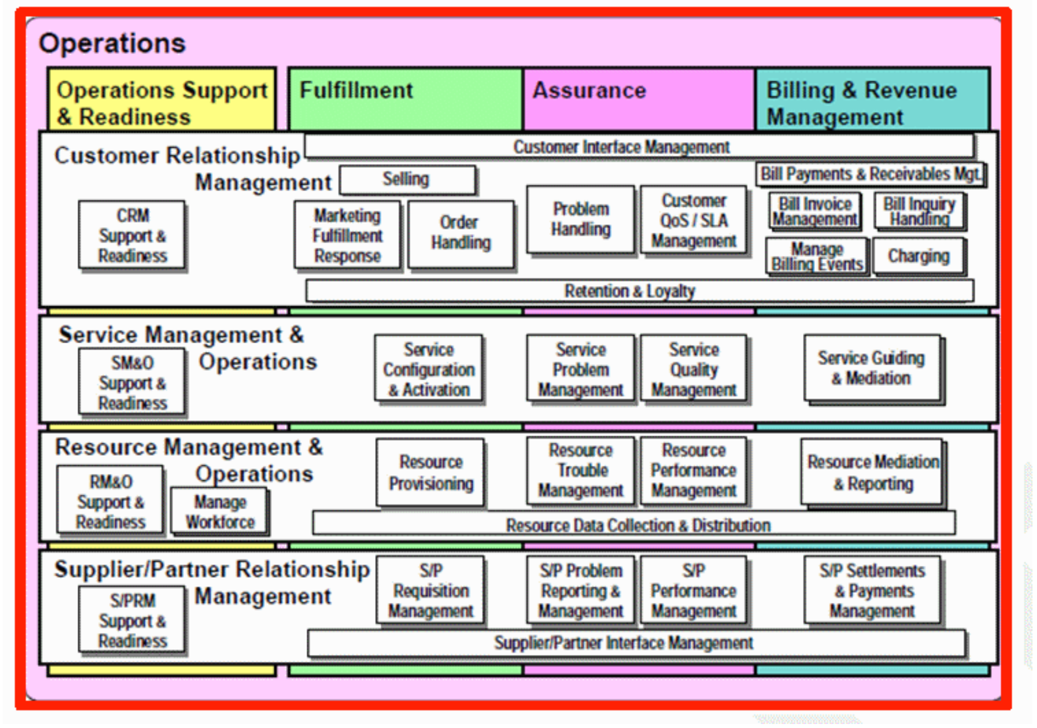

Operations
Operations Support & Readiness
Fulfillment
Assurance
Billing & Revenue Management
Customer Relationship Management
Customer Interface Management
Selling
Bill Payments & Receivables Mgt.
CRM Support & Readiness
Marketing Fulfillment Response
Order Handling
Problem Handling
Customer QoS / SLA Management
Bill Invoice Management
Bill Inquiry Handling
Manage Billing Events
Charging
Retention & Loyalty
Service Management & Operations
SM&O Support & Readiness
Service Configuration & Activation
Service Problem Management
Service Quality Management
Service Guiding & Mediation
Resource Management & Operations
RM&O Support & Readiness
Manage Workforce
Resource Provisioning
Resource Trouble Management
Resource Performance Management
Resource Mediation & Reporting
Resource Data Collection & Distribution
Supplier/Partner Relationship Management
S/PRM Support & Readiness
S/P Requisition Management
S/P Problem Reporting & Management
S/P Performance Management
S/P Settlements & Payments Management
Supplier/Partner Interface Management

Figure 5.1: eTOM Level 2 functionalities considered by VOSS

The Operations, Support and Readiness functionality ensures that the operational environment for the Fulfilment, Assurance, and Billing and Revenue Management (FAB) functionality is in place.

Operational management has a direct relation to the business model and thus possibilities regarding the business model and a choice are presented. Some of the business aspect definitions used in virtualised environments have been adopted from GEYSERS deliverable D1.1 "Identification, Description and Evaluation of the Use Case Portfolio and Potential Business Models" [GEYSERS-D1.1]; other definitions are unique to GENUS. The GEYSERS RORA model ([GEYSERS-D1.1], Section 2.2) is used to discuss the entities involved in the usage and operational management of resources.

The plan is for JRA1 Task 4 to integrate the GN3 NMA and the GEYSERS RORA model as part of the GENUS documentation, as the RORA model proposed is not only applicable to the management of resources.

5.2 RORA model

GEYSERS ([GEYSERS-D1.1], Section 2.2) defines "a novel business model that allows us to describe the different elements and their relationships, the so-called RORA Model, which takes its name from the four components it is based on: Resources, Ownership, Roles, and Actors. We base the RORA model on business scenarios where the vertical disintegration has led to [the sharing of] a resource substrate among the different involved entities; although each one of them holds a different set of allowed actions over it. These rights and responsibilities are defined by an agreement with another entity; and by establishing this new agreement we say that the entity obtains a specific ownership over the resource.”

5.2.1 Applicability of the RORA model in the NREN environment

The GEYSERS RORA model and its parts come from a different, more business-oriented environment than the NREN community. However, with regard to the operational management aspects, the NREN environment is not that different from the GEYSERS environment. The additions to the GENUS version of the model proposed below incorporate ideas utilised in NRENs (such as HEAnet).

5.2.2 Resources

A service consists of several aspects relating to Net(work) or IT resources:

- Worker Resource (WoR): the part actually used by the consumer; such as connectivity, computation, storage functions.
- Management Resource (MaR): the part that relates to the operational management of a WoR.

A Resource can also be a machine or a human (Human Resource (HR)).

The Human Resource is very flexible and can do a lot of things, including manual activities. Amazon’s Mechanical Turk [MTurk] is an example how to use HRs in a Web Services environment. Like a machine resource, an HR can be utilised as a WoR or a MaR.

In general, Resources can be composed together (aggregated) or they can be partitioned, as shown in Figure 5.2.

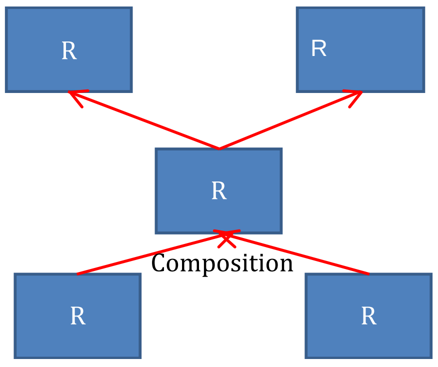


Figure 5.2: Resource aggregation and partitioning

5.2.2.1 *Worker Resources*

GEYSERS

In GEYSERS, a resource can have several types, depending on the virtualisation degree ([GEYSERS-D1.1] pp. 14–15, updated in [GEYSERS-D3.1] pp. 24–25):

- Physical Resource (PR): a real physical box (no virtualisation).
- Logical Resource (LR): a data model created from the PR, that is, an abstract representation of the PR (resource attributes are virtualised).
- Virtual Resource (VR): a logical construct behaving like a PR. It can be a simple abstraction, a partition of a PR or an aggregation of several PRs, but presented as a standalone resource (resource attributes are virtualised and manipulated, and new configuration/management interfaces are created).
- Virtual Infrastructure (VI): a composition of multiple VRs together.

GENUS

Like GEYSERS, GENUS has one overall Resource (seen as the object). A Resource can be subdivided into several types:

- Physical Resource (PR): a real physical box.
- Virtual Resource (VR): a partition of a physical box (PR), but presented as a standalone resource.
- Virtual Infrastructure (VI): a composition of multiple VRs together.

In this document, these Worker Resource types are seen as equivalent. A VI is a composition of many VRs. A VR will be built on a PR (or a VR). The HR is a very special PR. In essence, however, they are all a Worker Resource, and share the properties of a Worker Resource. (Such differences as there are relate to Service Level Specification (SLS), complexity, the extent to which the Resource is related to a physical box, time to realise, etc.)

From the operational management point of view, there is no real difference between these types of Worker Resource (except perhaps a different level of need for HRs to support such management). In the rest of the document, therefore, the term Worker Resource is used to mean any of the abovementioned Worker Resource types.

5.2.2.2 *GENUS Management-Resource*

In addition to seeing a link, an interface, a router, a network or a disc as a Worker Resource, the related operational management functions of such a resource are also virtualised in GENUS in the Management Resource (MaR).

There are many MaRs and they can be composed (aggregated) or partitioned just like Worker Resources. Selected MaRs are mentioned as examples.

The Service Stratum management blocks described in the GN3 document “Definition of a Multi-Domain Control and Management Architecture” ([GN3-MJ2.1.1] Section 5.1) are convenient examples of Management Resources:

- Service Configuration and Activation (CoMaR): addresses installing new WoRs, configuring WoRs, collecting configuration data, optimising WoRs.
- Service Quality Management (QuMaR): addresses managing, tracking, monitoring and reporting on the performance of a specific WoR.
- Service Trouble Management (TrMaR): addresses recognising, isolating and correcting faults.
- Service Policy Management (PoMaR): addresses permissions to add, modify or delete policy rules and attributes with regard to the business logic.
- Service Information Management (InMaR): addresses maintaining WoR-specific data according to the Service Information model.

All these MaRs are essential for any individual and/or composed WoR and the MaRs are very tightly coupled with the WoR. One can see a set of MaRs as a kind of Customer Management Interface (CMI), but based on Web Services, and thus as any other Resource.

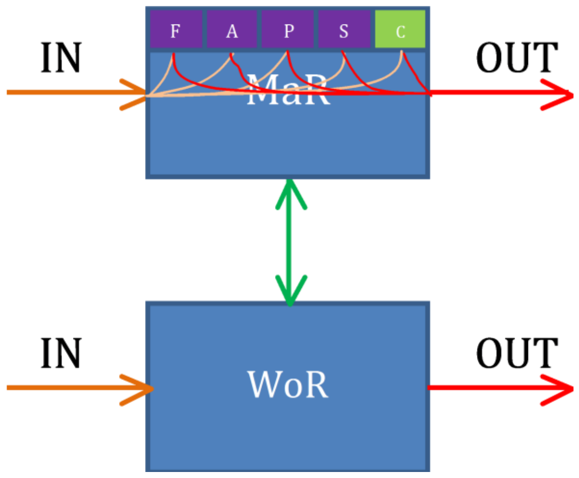


Figure 5.3: Relationship between WoR and MaR

Defining Service Stratum blocks as Web Service MaRs makes them easy to exchange and trade as if they were “normal” WoRs, thus providing a simpler service to the consumer who only wants to use a WoR (and perhaps not manage it). By virtualising and splitting the Service Stratum into different MaRs, a consumer has more control over to whom he/she can allocate a certain aspect of the management of a WoR (delegated responsibility).

The MaRs relate to the ability to perform the operational management function. The management tools themselves are provided by a role that provides the WoR and its related MaRs.

The MaRs can be implemented by HRs (certainly convenient if, for instance, physical installation is needed).

5.3 Ownership

GENUS follows GEYSERS in recognising five types of ownership ([GEYSERS-D1.1] p. 15):

1. Legal: the entity that purchased the PR and has legal responsibility for the actions performed with it.
2. Economic: the entity setting the policy.
3. Administrative: the entity that enforces the policy and performs certain management functions.
4. Operational: the entity that performs certain management functions and liaises with the service consumer.
5. Usage: the entity that uses the WoR or (sub)leases it to another service consumer.

Like GEYSERS, GENUS considers the legal and economic owners as the same entity, referred to as the economic owner.

The MaRs are related to the Administrative Owner (Tr-, In-, Qu-, Po-MaR) and Operational Owner (Co-MaR).

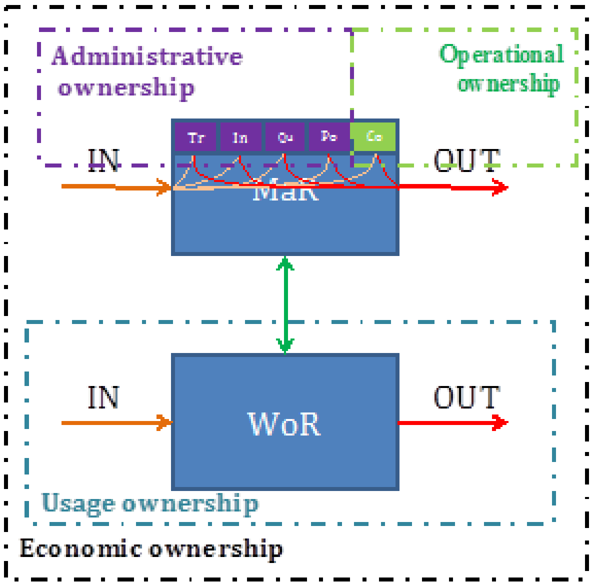


Figure 5.4: GENUS resource ownership model

5.4 Roles

This section presents the role structure used in GEYSERS and 4WARD, and defines the GENUS role structure.

5.4.1 GEYSERS

GEYSERS defines the following roles ([GEYSERS-D1.1] pp. 16–18):

- Physical Infrastructure Provider (PIP): has economic ownership of the equipment, and can lease PRs or VRs.
- Virtual infrastructure Provider (VIP): "Its main goal is to compose VRs belonging to [the same or] different PIPs in order to create VIs and offer them as a service towards the operator role." [GEYSERS-D1.1] (Administrative right). It is therefore a layer between the PIP and the VIO.
- Virtual Infrastructure Operator (VIO): controls the Resource in the operational phase (Operational right) and provides a service to the Service Consumer, built from the Resources leased by the PIP.
- Service Consumer (SC): the beneficiary of the services offered by the VIO; holds usage ownership only.

5.4.2 4WARD

4WARD uses a slightly different role structure, which was used in the MANTICORE II project and is due to be revised in the MANTYCHORE project:

- Infrastructure Provider (InP): maintains the PRs and enables the VRs.
- Virtual Network Provider (VNP): the provider that constructs from the Resources to create a Virtual Network (VNet) (or, in GEYSERS terminology, a VI).
- Virtual Network Operator (VNO): operates, controls and manages the VNet.

5.4.3 GENUS

The following roles are at present foreseen in the GENUS architecture:

- Service Consumer (SC) (close to GEYSERS' SC): the entity that uses the WoR and has the responsibility for allocating the related MaRs to Resource Operators.
- Resource Provider (RP) (close to 4WARD's InP and VNP and to GEYSERS' PIP): an entity that provides WoRs.
- Resource Operator (RO) (close to 4WARD's VNO and to GEYSERS' VIO): an entity that operationally manages part(s)/whole Resource through the MaRs.

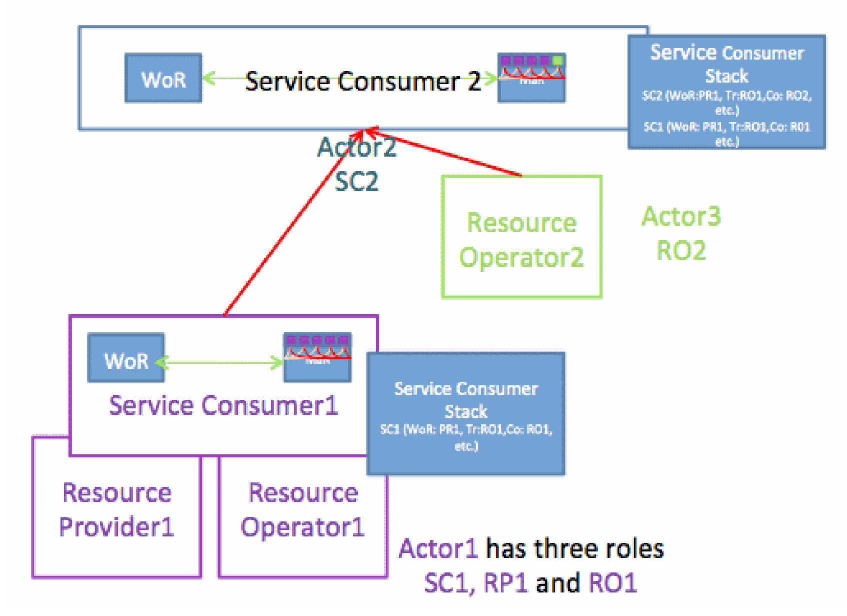


Figure 5.5: GENUS roles and actors

The GENUS roles have the advantage of being more generic and atomic than the GEYSERS roles and thus the options for combining roles (in actors – see Section 5.5 below) are more flexible and transparent.

As the principle of virtualisation allows sub-renting to another entity (depending of course on the policies set by the Economic Owner), a Service Consumer Stack (SCS), an integral property of a Resource, is being defined, which holds the SC and its related RPs and ROs. When a Resource is sub-rented to another SC, the new SC (and the operational management entities decided by the SC) will be pushed on the SCS.

5.5 Actors

An actor in GENUS is one or a combination of Roles, as in the GEYSERS RORA model. See Figure 5.5 above for an example of one role being performed by one Actor (Actor2 and Actor3) and an Actor having three roles (Actor1).

5.6 VOSS Pros and Cons

An operational GENUS mechanism to be deployed by NRENs requires the design and implementation of a comprehensive VOSS. This chapter has briefly discussed the considerations and required functionalities for such a VOSS. However, it must be noted that in addition to its many advantages, a VOSS also brings some disadvantages. Based on the discussion in this chapter, the pros and cons of a VOSS may be summarised as follows:

Pros

- Introduces a structured way to outsource operational management functions.
- Brings the operational management function into the Web Service domain.
- Re-uses a business model used for WoRs for MaRs, i.e. uses a unified model for both Resource types.
- As a result of the unified model, it is possible to use a similar concept like the marketplace.
- Introduces a standard protocol for accessing MaRs.
- The networking environment is moving towards a multi-tenant environment and standardising the API/interfaces for that environment is essential. VOSS contributes towards that standardisation.
- Results in the provision of a common management framework.

Cons

- Any unification layer normally adds complexity to the system.
- A multi-tenant environment poses extra risks to a system (see Chapter 3 *Drawback Analysis of Virtualisation*).
- Due to the present low level of availability of unified operational management services; VOSS is not easy to implement in the short term.

6 GENUS Prototype Design and Proof-of-Concept Implementation

6.1 Introduction

This chapter describes GENUS' software design, its implementation methodology and the current status of the GENUS prototype. Taking into account the architectural design of the GENUS system, a minimum set of functionality for GENUS proof-of-concept implementation has been identified, which is also described in this chapter.

6.2 Information modelling framework

To enable the different GENUS software components to interact using a common vocabulary, an information model needs to be in place. As such, the information model will be mostly an internal model for GENUS. JRA1 Task 4 has adopted the GEYSERS Information Modelling Framework (IMF), as described in GEYSERS deliverable D3.2. "Preliminary LICL software release" [GEYSERS-D3.2], from which much of the material in this section has been taken.

Figure 6.1 below shows the main hierarchy of the GEYSERS IMF as adopted by GENUS. All predicates or relations between the concepts are "is-a" relations. The top concept is a Resource, which can be a Device, a DeviceComponent or a NetworkElement. This enables devices, their components and the network elements that connect these devices to be described. Different types of device components exist, each with different properties. Memory, processing and storage components can be used to define IT resources. The operating system can be used to describe the platform of an IT resource. Switching components can be used to describe switches or routers. Specific types of optical switching components are also included, to describe their specific properties, which are needed for the virtualisation process of these optical components.

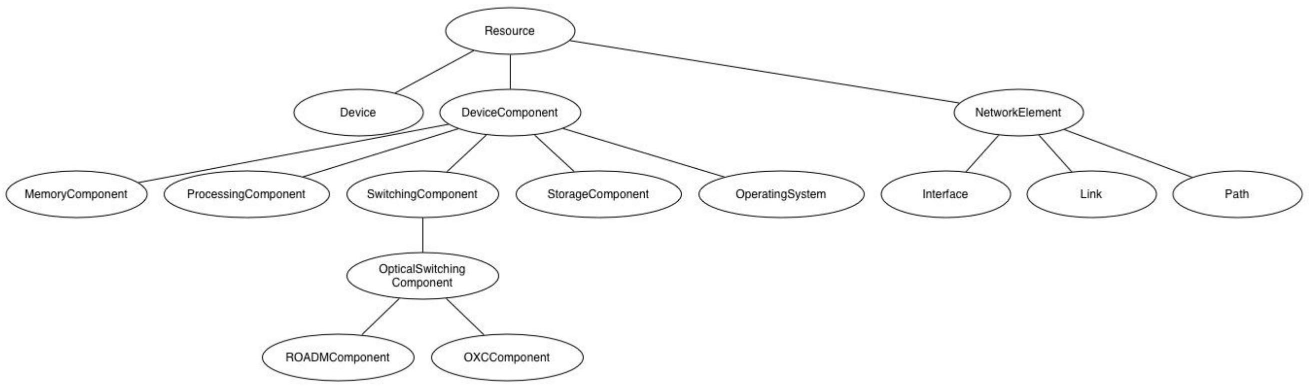


Figure 6.1: Main hierarchy of the IMF model

The network elements are defined in accordance with the current version of the Unified Modelling Language (UML) specifications. An interface enables the port via which a device is connected to another device to be described. Figure 6.2 below shows three different ways to connect two devices. (To distinguish between concepts in the IMF ontology, rectangular shapes are used to depict the instances of a concept.)

The first way of defining a connection is by creating a connectedTo predicate between two devices. This is the most abstract way of describing connections. The second way is by adding inbound and outbound interfaces to the devices and then connecting the interfaces of the two devices, which provides a more detailed description. The third and most detailed way of creating network connections is to connect interfaces using a link concept. The link concept contains properties such as the bandwidth of the link and the type of fibre (in the case of an optical link).

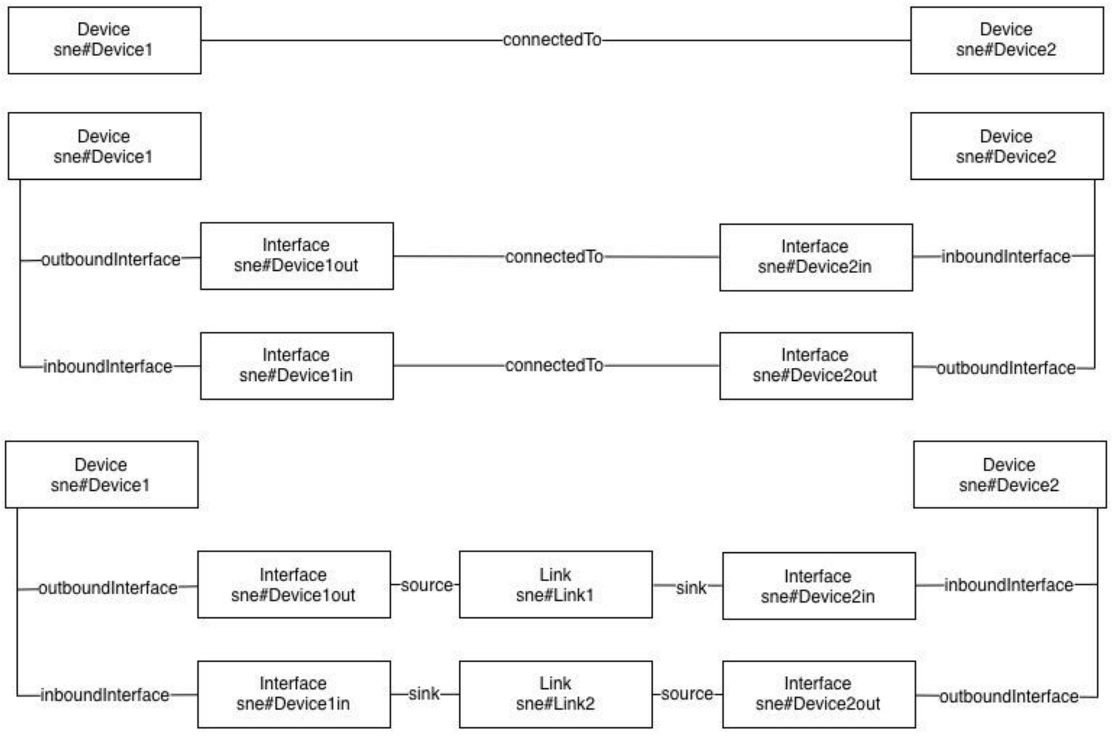


Figure 6.2: Different types of network connections

Figure 6.3 below shows the different properties of the Device concept. This concept can be used to describe physical and virtual devices as well as IT and network devices. Each device may consist of a number of components. If the device is a physical resource, it may consist of one or more processing components, memory and storage components. If the device is a virtual resource, the device description may only contain the component that characterises the device. As mentioned above, a device may have a number of inbound and outbound interfaces to connect it to other devices. Furthermore, a device can have a number of energy- and QoS-related properties. These properties are described in more detail in the following sections. A device can have an IPv4 address and/or an IPv6 address and/or a Universal Resource Indicator (URI). In the case of a physical device, the location of a device should correspond to the physical location of the device. In the case of a virtual device, the location may only indicate a city or country.

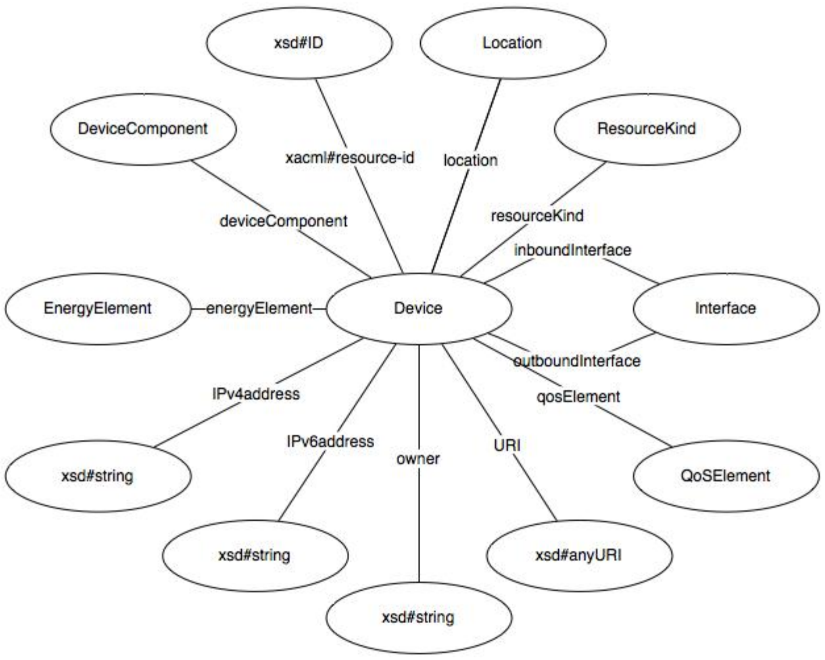


Figure 6.3: IMF Device properties

To define virtual infrastructures and virtual infrastructure requests, the Virtual Infrastructure concept is used. Figure 6.4 below shows all the properties of this concept. A virtual infrastructure consists of a number of resources, which can be devices, device concepts or network elements. To allow a virtual infrastructure to be defined in terms of device components, it is also possible to describe or request a virtual infrastructure with a certain storage or computing capacity without having to specify individual devices. Furthermore, a virtual infrastructure has a location to allow requests to be restricted to certain geographical areas. The virtual infrastructure may also have a number of energy- and QoS-related properties.

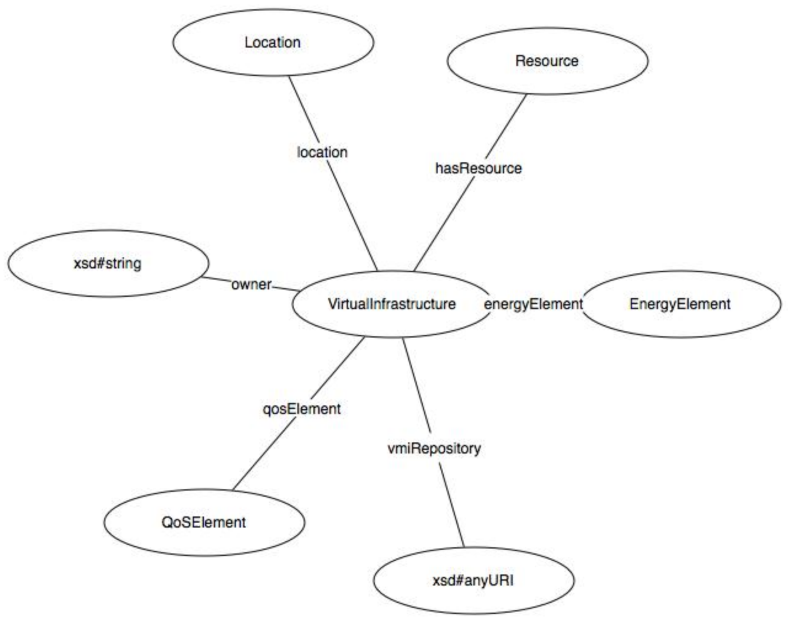


Figure 6.4: IMF Virtual Infrastructure properties

6.3 Implementation of virtual Infrastructure composition and operation

As mentioned above, taking into account the architectural design of the GENUS system, JRA1 Task 4 has identified a minimum set of functionality for GENUS prototype implementation. The following list describes the subset to be provided by the GENUS proof-of-concept implementation:

- Facility registration.
- Resource discovery.
- Requesting a virtual infrastructure.
- Decommissioning a virtual infrastructure.
- Booking resources.
- Releasing resources.

Figure 6.5, Figure 6.6 and Figure 6.7 present graphical representations of each function and the actors involved.

An overview of the status of the GENUS prototype based on the status of the functions and features is provided in Section 6.6.

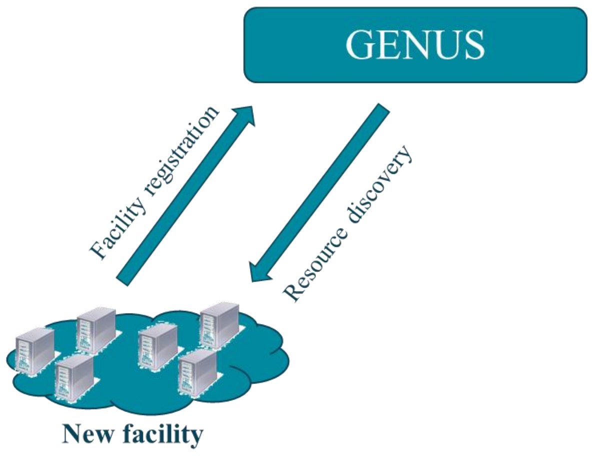


Figure 6.5: Registration of a new facility and resource discovery in the GENUS system

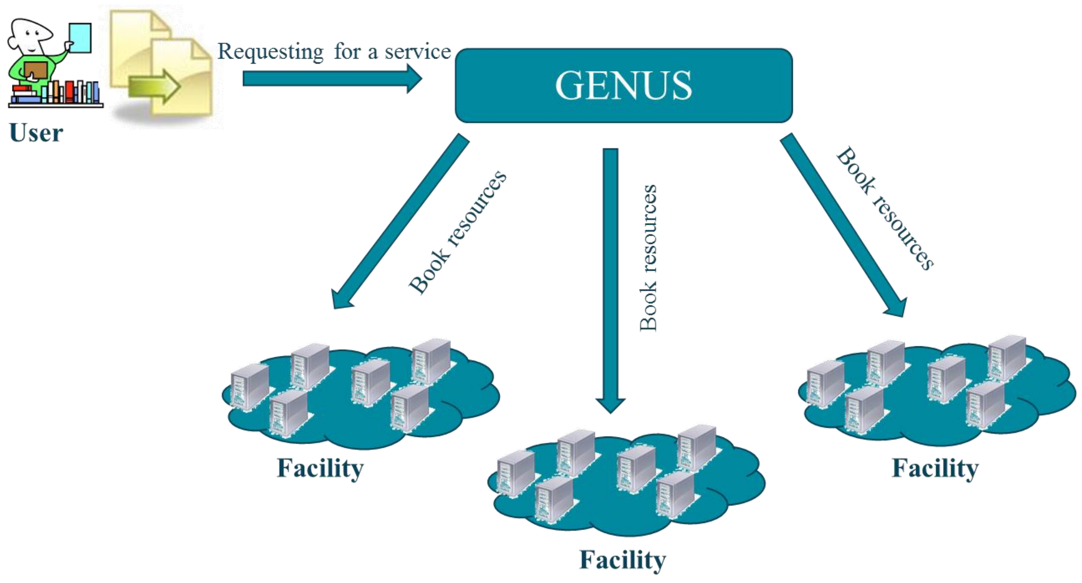


Figure 6.6: Requesting a new service (a new virtual infrastructure) from the GENUS system

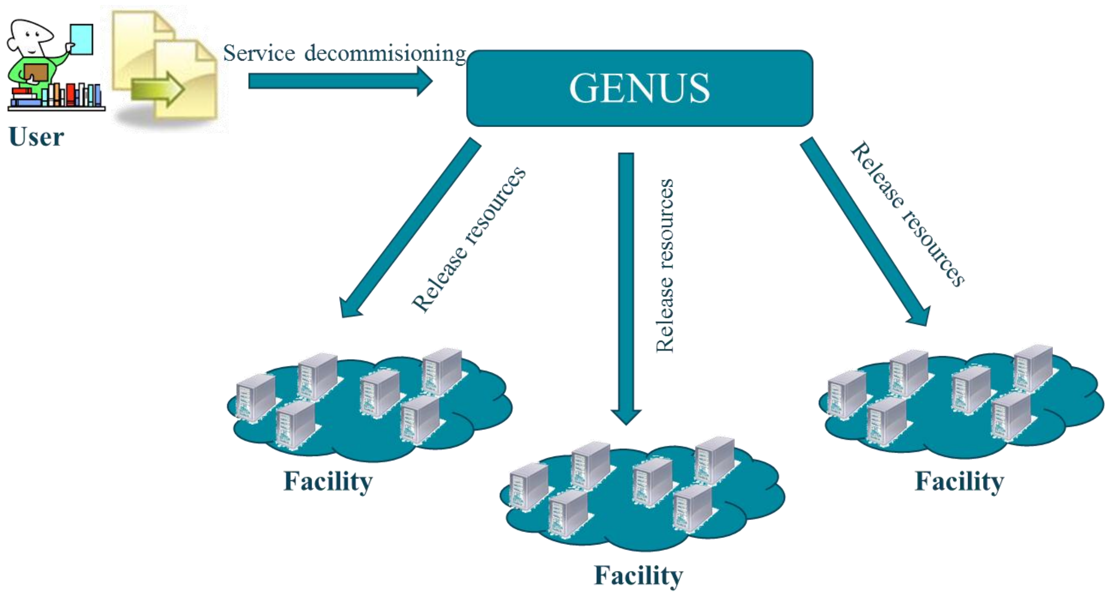


Figure 6.7: Decommissioning a service (a virtual infrastructure) in the GENUS system

The following sections describe each function in more detail, identifying the actors involved, the information exchanged and the implementation status.

6.3.1 Facility registration

In order to join the GENUS federation each facility is obliged to register itself in the GENUS registry. The registration procedure should cover exchange of basic information (e.g. site name, site administrator, site management platform, etc.). The advanced registration procedure should also cover methods for authentication and authorisation of users and accounting, which, although not being included in the proof of concept implementation, are acknowledged to be of high importance to the production service.

Actors involved:

- A new facility joining the federation.
- GENUS.

The following table summarises the information exchanged between the GENUS system and a new site during the facility registration phase.

Information	Provided by	Comments	
		Implementation considerations	Structure
New site name	New facility	Proof-of-concept implementation	Simple object – string

Information	Provided by	Comments	
		Implementation considerations	Structure
Site administrator	New facility	Proof-of-concept implementation	Composite object: • Admin name • Admin email address
Site management platform	New facility	Proof-of-concept implementation	Composite object: • Platform name • Access method (Web Services, Corba, other) • Access URL
AAA information	New facility	Advanced feature (not considered in the proof-of-concept implementation)	Composite object: • Authentication & Authorisation method • AA system URL
Accounting	New facility	Advanced feature (not considered in the proof-of-concept implementation)	Composite object: • Access URL
Registration confirmation	GENUS	Proof-of-concept implementation	Composite object: • GENUS site Id

Table 6.1: Information exchange during facility registration

Implementation of this feature has been finalised and it is ready for GENUS integration.

6.3.2 Resource discovery

Each facility has to inform the GENUS system about the resources available for use in a federation. This can be realised through either a pooling or a notification mechanism. In both cases the facility participating in a federation provides information about its resources. The only difference is who triggers the procedure: the GENUS system or the facility itself. This decision will be taken by the implementation team during the design phase of the GENUS system.

Actors involved:

- A new facility joining the federation.
- GENUS.

The following table summarises the information exchanged between the GENUS system and a new site during the resource discovery phase.

Information	Provided by	Comments	
		Implementation considerations	Structure
Resource discovery request	GENUS	Proof-of-concept implementation	An empty request
Resources list	New facility	Proof-of-concept implementation	Composite object: • List of resources available in the testbed
Resource discovery confirmation	GENUS	Proof-of-concept implementation	An empty request

Table 6.2: Information exchange during resource discovery

This feature will not be available for the first release of the GENUS prototype.

6.3.3 Requesting a virtual infrastructure from the GENUS system

The end user may request a virtual infrastructure from the GENUS system. The request will be further processed by internal components and decomposed into a set of requests passed to each facility participating in a service (more information is provided in Section 6.3.5 *Booking resources in local facilities*). The current implementation of the GENUS system allows the end user to request a virtual infrastructure using two methods:

- By describing the virtual infrastructure based on the GEYSERS IMF mentioned above and in accordance with UML specifications.
- By browsing through the available facilities and registered resources within the GENUS system and selecting the required services/resources.

Actors involved:

- A user requesting a virtual infrastructure.
- GENUS.

The following table summarises the information exchanged between the user requesting a service (virtual infrastructure) and the GENUS system.

Information	Provided by	Comments	
		Implementation considerations	Structure
User information	User	Proof-of-concept implementation	Composite object: • User name • User email address

Information	Provided by	Comments	
		Implementation considerations	Structure
AA information	User	Advanced feature (not considered in the proof-of-concept implementation)	Composite object: • Authentication & Authorisation method • AA system URL
Service (virtual infrastructure) specification	User	Proof-of-concept implementation	Composite object: • Specific details of the service (dates, end points, list of virtual/physical resources, etc.)
Service (virtual infrastructure) status	GENUS	Proof-of-concept implementation	Composite object: • Service Id • Service status

Table 6.3: Information exchange during service request (virtual infrastructure)

Implementation of this feature has been finalised and it is ready for GENUS integration.

6.3.4 Decommissioning a virtual infrastructure in the GENUS system

It is expected that the user will specify the start and end date of the service (virtual infrastructure) in the request message. However, it is possible that the user may want to terminate the service (virtual infrastructure) before it decommissions automatically.

Actors involved:

- A user requesting a service (virtual infrastructure) termination.
- GENUS.

The following table summarises the information exchanged between the user requesting a service (virtual infrastructure) termination and the GENUS system.

Information	Provided by	Comments	
		Implementation considerations	Structure
AA information	User	Advanced feature (not considered in the proof-of-concept implementation)	Composite object: • Authentication & Authorisation method • AA system URL
Service (virtual	User	Proof-of-concept	Simple object – Service Id

Information	Provided by	Comments	
		Implementation considerations	Structure
infrastructure) Id		implementation	
Service (virtual infrastructure) status	GENUS	Proof-of-concept implementation	Composite object: • Service Id • Service status

Table 6.4: Information exchange during service termination (virtual infrastructure)

Implementation of this feature has been finalised and it is ready for GENUS integration.

6.3.5 Booking resources in local facilities

The GENUS system is responsible for running advanced algorithms to find the optimal allocation of resources in participating domains in the federation.

The complex request for a service (virtual infrastructure) is decomposed into a set of requests for allocation of resources in independent management systems running on top of each domain.

Actors involved:

- GENUS.
- A facility participating in a service (virtual infrastructure).

The following table summarises the information exchanged between the GENUS system and a facility participating in a service to optimise resources.

Information	Provided by	Comments	
		Implementation considerations	Structure
Service (virtual infrastructure) information	GENUS	Proof-of-concept implementation	Composite object: • Service Id • Service description • Start/end date
Reservation information	GENUS	Proof-of-concept implementation	Composite object: • Resource list • Start/end date
Reservation status	Facility	Proof-of-concept implementation	Composite object: • Reservation Id

Information	Provided by	Comments	
		Implementation considerations	Structure
			• Reservation status • Is GENUS responsible for releasing resources or it will be done automatically?

Table 6.5: Information exchange during service participation – resource optimisation

Implementation of this feature has been finalised and it is ready for GENUS integration.

6.3.6 Releasing resources

If the service (virtual infrastructure) is terminated manually by a user or if the facility explicitly requested a release date during the reservation phase, the GENUS system is responsible for triggering a set of requests to particular facilities participating in a service for releasing resources. For more information, please refer to Section 6.3.5 *Booking resources in local facilities* above.

Actors involved:

- GENUS.
- A facility participating in a service (virtual infrastructure).

The following table summarises the information exchanged between the GENUS system and a facility participating in a service to release resources.

Information	Provided by	Comments	
		Implementation considerations	Structure
Service (virtual infrastructure) information	GENUS	Proof-of-concept implementation	Composite object: • Service Id • Service description • Start/end date
Reservation Id	GENUS	Proof-of-concept implementation	Simple object – reservation Id
Reservation status	Facility	Proof-of-concept implementation	Composite object: • Reservation Id • Reservation status

Table 6.6: Information exchange during service participation – resource release

Implementation of this feature has been finalised and it is ready for GENUS integration.

6.4 Implementation of NREN and GÉANT adaptors

The GENUS prototype aims to support MANTYCHORE (IP/L2 virtualisation) and OFELIA (OpenFlow) virtualisation systems as well as the GÉANT AutoBAHN tool. Development of adaptors for MANTYCHORE and OFELIA is in its final stage (a test and debugging period) while development of the adaptor for AutoBAHN has been finalised and its prototype is ready for GENUS integration.

6.5 Implementation of unified user interface

The GENUS unified user interface has been implemented and its first prototype is ready for integration with the GENUS system. The current user interface supports the following functionalities:

- Manual facility and resource registration.
- Resource listing.
- Virtual infrastructure request submission.

The figures below show screenshots of the GENUS unified user interface.

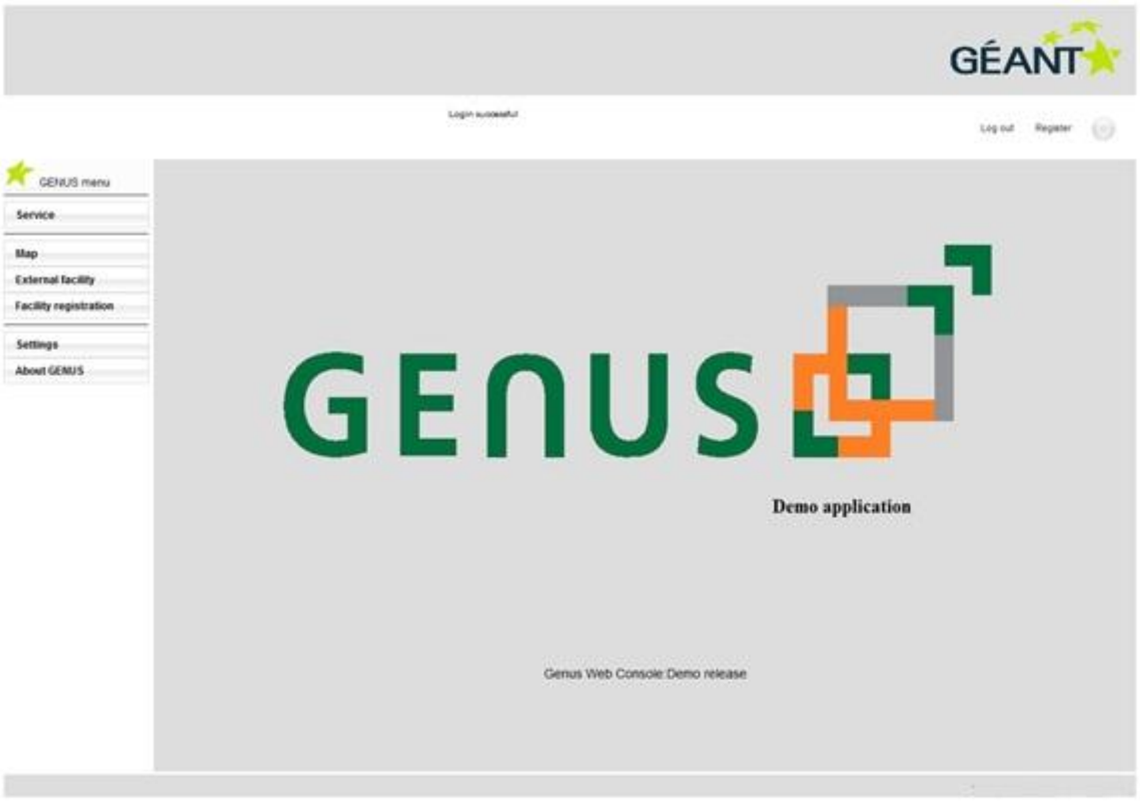


Figure 6.8: GENUS unified user interface: main menu

The user interface provides the following functionalities:

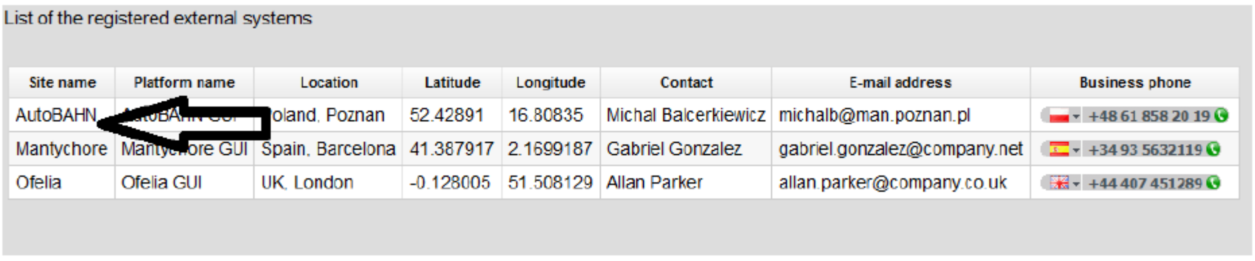
Service / Functionality	Description
Service (i.e. GENUS service)	Displays status of available infrastructures, their resources and their capability. It also allows the user to request and create virtual infrastructures using available resources.
Map	Provides a graphical representation of available resources and their capability.
External Facility	Allows the management of GENUS-registered infrastructures, their capabilities and their properties.
Facility registration	Allows the registering of an infrastructure or part of an infrastructure in the GENUS system.
Settings	Used for configuring the GENUS system.
About GENUS	Provides access to information about GENUS and its capability including help and user manual.

Table 6.7: GENUS user-interface functionality

6.5.1 GENUS virtualisation service interfaces

The GENUS virtualisation service (referred to in the user interface as Service) allows GENUS users to check the current status of the available resources and to browse through the available resources, infrastructures and their capability. It also provides the main service of GENUS, which is allowing users to request a virtual infrastructure in the GENUS system.

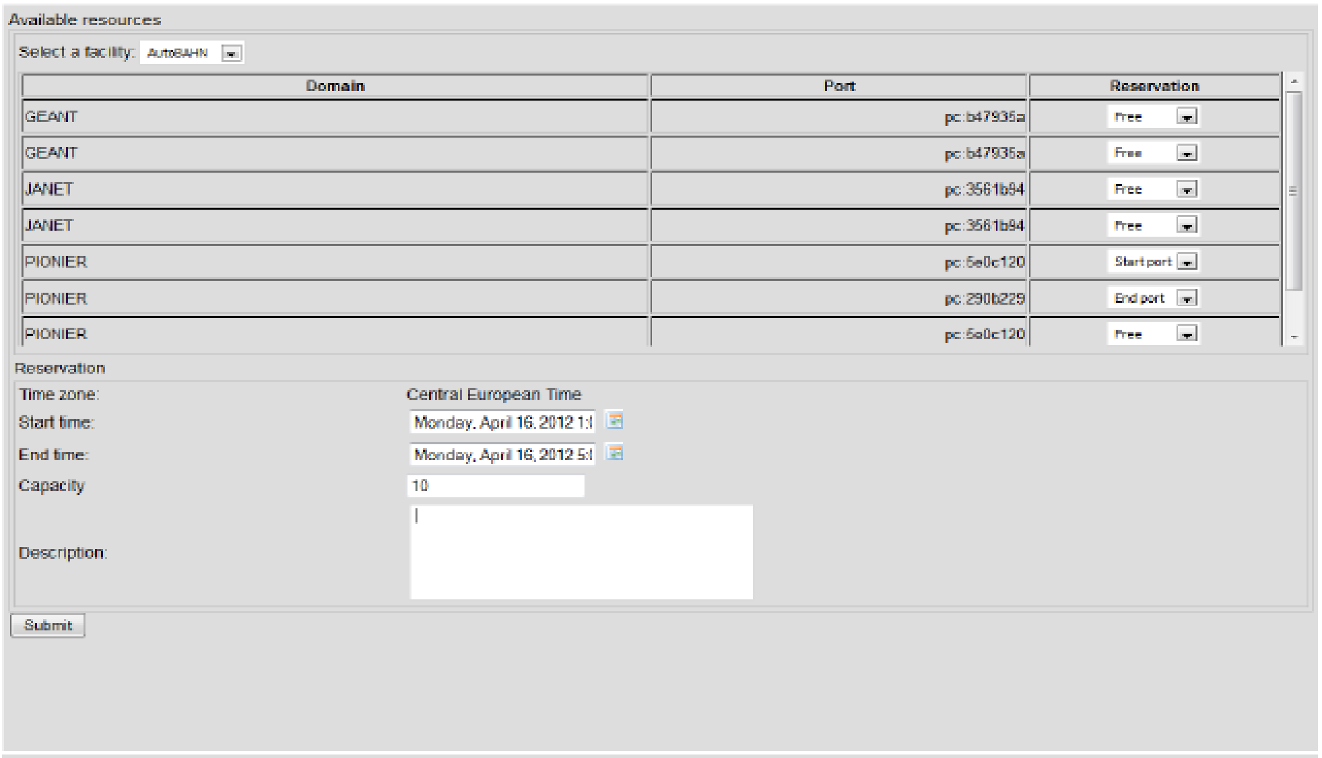
Figure 6.9 shows a list of external registered facilities in the GENUS system, which is accessible via the External Facility menu option. Users can browse through the registered infrastructures and facilities, and then choose one infrastructure and browse through its available resources.



List of the registered external systems

Site name	Platform name	Location	Latitude	Longitude	Contact	E-mail address	Business phone
AutoBAHN	AutoBAHN GUI	oland, Poznan	52.42891	16.80835	Michal Balcerkiewicz	michalb@man.poznan.pl	+48 61 858 20 19
Mantychore	Mantychore GUI	Spain, Barcelona	41.387917	2.1699187	Gabriel Gonzalez	gabriel.gonzalez@company.net	+34 93 5632119
Ofelia	Ofelia GUI	UK, London	-0.128005	51.508129	Allan Parker	allan.parker@company.co.uk	+44 407 451289

Figure 6.9: GENUS unified user interface: list of registered external systems



Available resources

Select a facility: AutoBAHN

Domain	Port	Reservation
GEANT	pc:b47935a	Free
GEANT	pc:b47935a	Free
JANET	pc:3561b94	Free
JANET	pc:3561b94	Free
PIONIER	pc:5e0c120	Start port
PIONIER	pc:290b229	End port
PIONIER	pc:5e0c120	Free

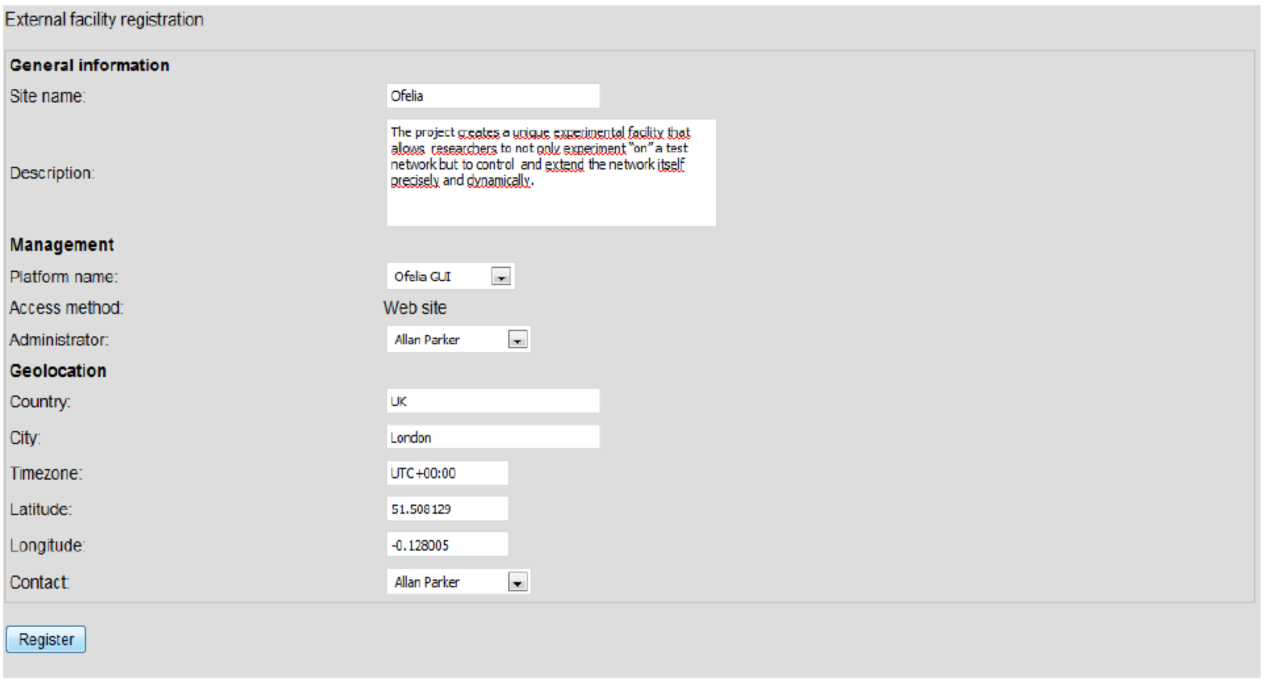
Reservation

Time zone:	Central European Time
Start time:	Monday, April 16, 2012 1:(
End time:	Monday, April 16, 2012 5:(
Capacity	10
Description:	

Figure 6.10: GENUS unified user interface: service for browsing available resources of an infrastructure or facility

6.5.2 Facility registration interface

This service allows an infrastructure owner (NREN) who is willing to share its infrastructure (or part of its infrastructure) to register its facility in the GENUS system. For an infrastructure to be registered in GENUS, it has to deploy (fully or partially) one of the virtualisation mechanisms supported by GENUS (currently MANTYCHORE and OFELIA) or Autobahn.



External facility registration

General information

Site name:	Ofelia
Description:	The project creates a unique experimental facility that allows researchers to not only experiment "on" a test network but to control and extend the network itself precisely and dynamically.

Management

Platform name:	Ofelia GUI
Access method:	Web site
Administrator:	Allan Parker

Geolocation

Country:	UK
City:	London
Timezone:	UTC+00:00
Latitude:	51.508129
Longitude:	-0.128005
Contact:	Allan Parker

Figure 6.11: GENUS unified user interface: External facility registration menu

6.5.3 In development

Currently the GENUS development team is working on the implementation of the part of the user interface that allows users to reserve resources and create a virtual infrastructure. Once this has been finalised, the first prototype of GENUS will be ready for release.

6.6 First GENUS prototype release and current status

The first prototype of GENUS software integrating all the functionalities and features outlined above is scheduled for release at the end of June 2012. A final set of results will be documented in a white paper entitled "Full-Featured Virtualisation – Development, Demonstration and Use Cases", due to be available in March

2013. An overview of the current status of the GENUS prototype based on the status of the functions and features is provided in Table 6.8.

Function / Feature	Status
Facility registration	Implementation has been finalised. Ready for GENUS integration.
Resource discovery	Will not be available for the first release of the GENUS prototype.
Requesting a virtual infrastructure	Implementation has been finalised. Ready for GENUS integration.
Decommissioning a virtual infrastructure	Implementation has been finalised. Ready for GENUS integration.
Booking resources	Implementation has been finalised. Ready for GENUS integration.
Releasing resources	Implementation has been finalised. Ready for GENUS integration.
NREN & GÉANT adaptors: • MANTYCHORE • OFELIA • AutoBAHN	 • In final development stage (test and debugging period). • In final development stage (test and debugging period). • Finalised, Ready for GENUS integration.
Unified user interface: • Service • Map • External Facility • Facility registration • Settings • About GENUS	 • Nearly complete. • Complete. • Complete. • Complete. • Complete. • Complete.

Table 6.8: Overview of current status of GENUS prototype

7 GENUS Testbed

The GENUS testbed is comprised of resources kindly committed by the JRA1 Task 4 partners. Each partner provides local facilities, which will be interconnected together either via dynamic network services offered by the GÉANT core network or through the FEDERICA infrastructure.

Figure 7.1 presents a logical view of the GENUS testbed architecture.

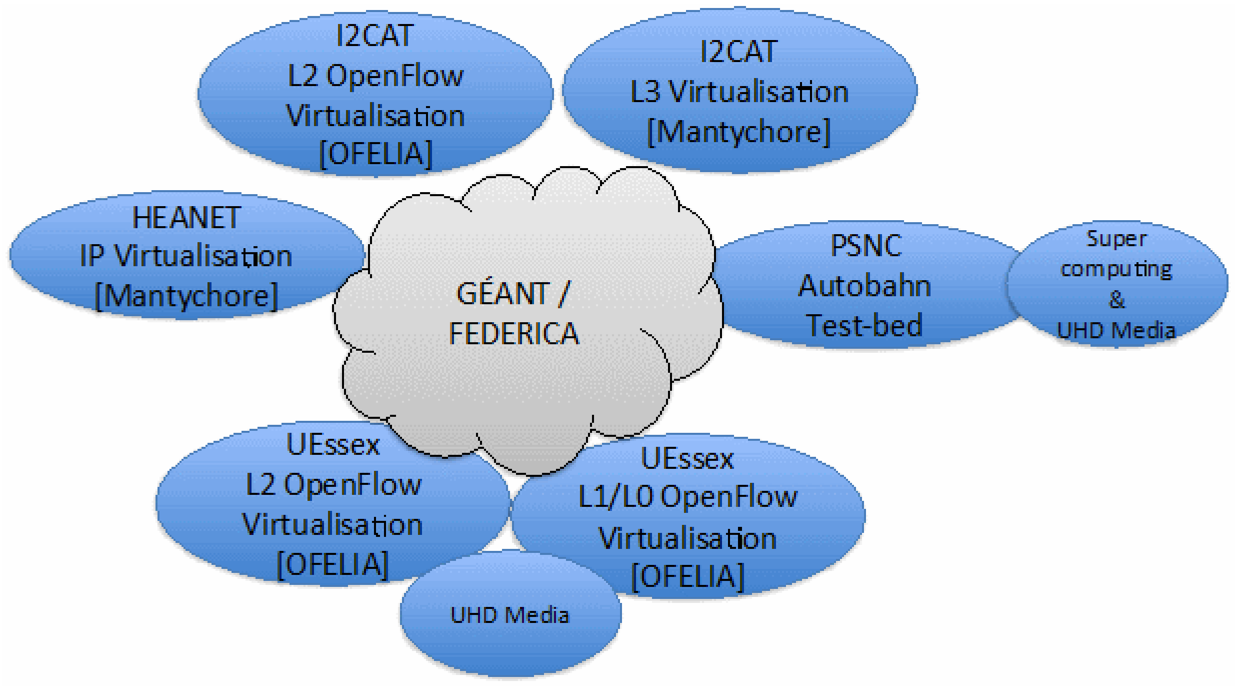

I2CAT L2 OpenFlow Virtualisation [OFELIA]
I2CAT L3 Virtualisation [Mantychore]
HEANET IP Virtualisation [Mantychore]
GÉANT / FEDERICA
PSNC Autobahn Test-bed
Super computing & UHD Media
UEssex L2 OpenFlow Virtualisation [OFELIA]
UEssex L1/L0 OpenFlow Virtualisation [OFELIA]
UHD Media

Figure 7.1: Logical view of the GENUS testbed architecture

This chapter provides details of the testbed capabilities and describes how the project partners will be interconnected to create a distributed GENUS testing environment.

7.1 Testbed backbone

The testbed backbone will be used to interconnect project partners' facilities. It will be based either on the capabilities of services offered by the GÉANT core network or on the FEDERICA infrastructure.

7.1.1 FEDERICA infrastructure

Although the FEDERICA project itself has been terminated, the infrastructure is still in place and operational. Currently all Network Operations Centre-related functions are handled by a team set up in the Poznań Supercomputing and Networking Centre (PSNC). It has been agreed that the infrastructure can be used by external researchers (e.g. GN3 project members) to perform disruptive or non-disruptive experiments. It has also been specifically agreed that it will support GENUS demonstration activity.

7.1.2 Dynamic network services over GÉANT

In addition to using the FEDERICA infrastructure, GENUS will rely on existing GÉANT connectivity between testbed partners. Currently all partners are connected to PSNC via GÉANT.

7.2 Local facilities attached to the testbed backbone

7.2.1 IP network infrastructure

HEAnet and i2CAT have offered their Layer 3 infrastructure for building the GENUS validation environment. The infrastructure comprises a number of virtual IP routers (the exact number will be defined soon) and a control framework – the MANTYCHORE software suite– to manage the physical/virtual infrastructure at HEAnet and i2CAT.

7.2.2 OpenFlow testbeds

i2CAT and the University of Essex will construct a multi-layer, multi technology OpenFlow-enabled testbed comprising of a Dense Wavelength-Division Multiplexed (DWDM) optical network domain, a Carrier Grade Ethernet network domain and a campus Ethernet domain. The DWDM optical domain comprises of three ADVA optical switching nodes. The team at Essex, in collaboration with ADVA, is developing a Layer 1 OpenFlow controller for ADVA switches enabling partitioning (virtualisation) of the WDM for concurrent and independently controlled Layer 1 network experiments.

The Carrier Grade Ethernet domain comprises of three OpenFlow-enabled Extreme carrier-class switching nodes and the campus Ethernet domain comprises of four OpenFlow NEC Ethernet switches.

7.2.3 AutoBAHN testbed

PSNC will offer an AutoBAHN testbed for GENUS prototype testing which will emulate the GÉANT BoD service.

7.2.4 Media laboratories

In 2008 PSNC began the creation of a 4K node in Poznań. It consists of devices that enable the recording, storing, projecting and network streaming of movies with 4K resolution. In addition to the node offered by PSNC, the University of Essex will provide access to very advanced ultra-high definition video sources (4K 3D and 8K) which can generate high bit rate data streams (up to 20 Gbit/s) and be used as test applications for experiments carried out over the facility. This will be used as the application test for the GENUS prototype demonstration.

7.3 First GENUS prototype demonstration

The first demonstration of the GENUS software prototype over the testbed outlined above is scheduled for the end of June 2012. It will demonstrate the subset of functionality described in Chapter 6 and GENUS's capability to create virtual infrastructure using resources from several testbeds. A final set of results and definition of success criteria will be documented in a white paper entitled "Full-Featured Virtualisation – Development, Demonstration and Use Cases", due to be available in March 2013.

8 Conclusions

This deliverable has reported on a comparative study of several major infrastructure virtualisation frameworks, both existing and under development, in Europe, USA and Japan. From the findings of this study it is evident that the European research community, helped by the drive and commitment of the NRENs, has achieved significant progress on infrastructure virtualisation technologies through projects such as OFELIA, MANTYCHORE, NOVI and GEYSERS. These projects are complementary and, combined together, they can provide virtualisation of Layer 1, Layer 2 and Layer 3 networks as well as computing resources.

JRA1 Task 4 has therefore focused on adopting the successful outcomes of these projects for a GÉANT virtualisation service. Rather than proposing a specific virtualisation technology, the Task proposes an integrated architecture approach that allows the different virtualisation technologies deployed across the NRENs and GÉANT to be integrated, offering a multi-layer, multi-domain and multi-technology virtualisation service. This approach enables each NREN to adopt one or multiple virtualisation technologies of their choosing, depending on their requirements, and to offer to its users inter- and/or intra-domain as well as multi-layer infrastructure virtualisation services.

This vision is realised through the proposed GENUS multi-layer, multi-domain virtualisation system for GÉANT and its associated NRENs. The report has discussed the required functionality for operation support and service of a virtualised infrastructure from both an operator and user point of view, as well as relevant issues.

The results of a drawback analysis of virtualisation deployment for NRENs and GÉANT were presented. The analysis concluded that there are no major issues with regard to the technical features required for the hardware and software to provide the necessary capabilities for virtualisation. However, apart from these purely technical aspects, somewhat larger problems still exist, especially with regard to the operational environment, the maturity of solutions and the area of security.

All these elements offer guidance with regard to virtualisation services in the GÉANT community, at the same time as acknowledging the different requirements of the members of that community.

Finally, the report has described JRA1 Task 4's prototype implementation of GENUS as well as the deployment of a multi-domain and multi-layer virtualisation testbed for future improvement and investigation of issues relevant to GENUS and the GÉANT virtualisation service.

References

[AKARI-ConceptualDesign]	"New Generation Network Architecture: AKARI Conceptual Design" http://akari-project.nict.go.jp/eng/concept-design/AKARI_fulltext_e_preliminary_ver2.pdf
[Argia]	E. Grasa, S. Figuerola, A. Forns, G. Junyent, J. Mambretti, "Extending the Argia Software with a Dynamic Optical Multicast Service to support High Performance Digital Media", accepted for publication in Elsevier journal of Optical Switching and Networking Volume 6, Issue 2, April 2009 http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B7GX5-4VXMPRK-1&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&_docanchor=&view=c&_searchStrId=1077462195&_rerunOrigin=google&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=d6a9240fe4221f8d93569fc5868870be
[Ether]	[details of reference to be provided]
[Expedient]	http://yuba.stanford.edu/~jnaous/expedient/docs/admin/install.html
[FEDERICA-DSA1.1]	Deliverable "DSA1.1: "FEDERICA Infrastructure" http://www.fp7-federica.eu/documents/FEDERICA-DSA1.1.pdf
[FlowVisor]	Rob Sherwood, et al., "FlowVisor: A Network Virtualization Layer", Oct 2009 www.openflow.org
[FlowVisor2]	http://openflowswitch.org/wk/index.php/FlowVisor
[Fuse]	Fuse ESB http://fusesource.com/products/enterprise-servicemix/
[GEMBus]	Diego R. Lopez, "The GEMBus Way: Delivering the Promise of the Internet of Services" http://www.terena.org/activities/eurocamp/nov09/slides/20091118-GEMBus-EuroCAMP-Budapest.ppt
[GENI]	http://www.geni.net
[GENI-GDD-07-44]	L. Peterson (ed.), "GENI: Global Environment for Network Innovations – Facility Design", GDD-07-44, March 2007
[GENI-Intro]	Harry Mussman, "GENI: An Introduction", 2011, GENI WIKI
[GENI-Overview]	"The Global Environment for Network Innovations (GENI)", April 2009 http://www.geni.net/wp-content/uploads/2009/04/geni-at-a-glance-final.pdf
[GENI-SFA]	Larry Peterson, Robert Ricci, Aaron Falk, Jeff Chase "Slice-Based Federation Architecture", 2010, GENI WIKI
[GENI-System]	[details to be provided]
[GEYSERS-D1.1]	GEYSERS deliverable D1.1 "Identification, Description and Evaluation of the Use Case Portfolio and Potential Business Models" http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_1.1.pdf

[GEYSERS-D2.1]	GEYSERS deliverable D2.1 “Initial GEYSERS Architecture and Interfaces Specification” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_2.1.pdf
[GEYSERS-D3.1]	GEYSERS deliverable D3.1 “Functional Description of the Logical Infrastructure Composition Layer (LICL)” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_3.1.pdf
[GEYSERS-D3.2]	GEYSERS deliverable D3.2 “Preliminary LICL software release” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_3.2-v1_final.pdf
[GEYSERS-D4.1]	GEYSERS deliverable D4.1 “GMPLS+/PCE+ Control Plane Architecture” http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_4.1.pdf
[GN3-DJ1.4.1]	M. Campanella, P. Kaufman, F. Loui, R. Nejabati, C. Tziouvaras, D. Wilson, S. Tyley, “Deliverable DJ1.4.1: Virtualisation Services and Framework Study” http://www.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-09-225%20DJ1.4.1v1.0%20Virtualisation%20Services%20and%20Framework%20Study.pdf
[GN3-DJ2.1.1]	A. Sevasti, P. Vuletic, D. Kalogeras, M. Giertych, D. Parniewicz, D. Pajin, “Information Schemas and Workflows for Multi-Domain Control and Management Functions [restricted access]
[GN3-MJ2.1.1]	“Definition of a Multi-Domain Control and Management Architecture” https://intranet.geant.net/sites/Research/JRA2/T1/Documents/NMAv1.0.1.doc [PP - access restricted to project participants]
[GoogleAEBlog]	http://googleappengine.blogspot.com/
[Google-Intro]	http://code.google.com/appengine/docs/whatisgoogleappengine.html
[IaaS]	Infrastructure as a Service http://www.iaasframework.com
[MANTYCHORE]	MANTYCHORE website http://www.mantychore.eu/
[MANTYCHORE-DoW]	MANTYCHORE “Description of Work” http://jira.i2cat.net:8090/download/attachments/3211820/MANTYCHORE+FP7+-+DoW+-+Part+B+-+final+-+budget+removed.pdf
[MTurk]	https://www.mturk.com/mturk/welcome
[Nakao1]	Akihiro Nakao, “Network Virtualization as Foundation for Enabling New Network Architectures and Applications”, IEICE Transactions on Communications, Volume E93.B, Issue 3, pp. 454-457 (2010) adsabs.harvard.edu/abs/2010IEITC..93..454N
[Nakao2]	Akihiro Nakao, “Architectures and Testbeds Enabled Through Advanced Network Virtualization: CoreLab and VNode Projects”, 3rd EU Japan Symposium on Future Internet http://ec.europa.eu/information_society/activities/foi/research/eu-japan/eujapan3/docs/nakao.pdf
[Nakauchi]	Kiyohide Nakauchi, “Introduction to Network Virtualization Technologies in Future Internet Research”, Asia FI Summer School (August 26, 2010) www.asiafi.net/meeting/2010/summerschool/p/nakauchi.pdf
[NDLRef]	https://noc.sara.nl/nrg/ndl/
[NICT]	National Institute of Information and Communications Technology, Japan
[NOVI]	http://www.fp7-novi.eu/
[NOVI-D3.1]	NOVI deliverable “D3.1 State-of-the-Art Management Planes” http://www.fp7-novi.eu/index.php/deliverables/doc_download/24-d31

[NOVI-D4.2]	NOVI Deliverable “D4.2: Use Cases” http://www.fp7-novi.eu/index.php/deliverables/doc_download/26-d42
[NOX]	www.noxrepo.org
[OFELIA]	http://www.fp7-ofelia.eu/
[OpenFlow1]	www.openflow.org
[OpenFlow2]	Nick McKeown, et al., “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication, Apr 2008
[Panlab]	http://www.panlab.net
[PlanetLab]]	http://www.planet-lab.org/
[Routing]	D. Wilson, “Routing Integrity in a World of Bandwidth on Demand”, TNC 2006 http://www.terena.org/events/tnc2006/programme/presentations/show.php?pres_id=242
[SFAImpl]	PlanetLab Implementation of the SFA https://svn.planet-lab.org/browser/sfa/trunk/docs/sfa-impl-2009-04-07.pdf
[SFAOver]	SFA Overview http://svn.planet-lab.org/wiki/SFAGuide#SFAOverview
[SFAv1]	“Slice-Based Facility Architecture”, Draft Version 1.04, April 7, 2009 https://svn.planet-lab.org/browser/sfa/trunk/docs/sfa.pdf
[SFAv2]	“Slice-Based Federation Architecture”, Version 2.0, July 2010 http://groups.geni.net/geni/wiki/SliceFedArch
[VMWARE]	http://www.vmware.com/
[VSERVER]	Virtualization for GNU/Linux systems http://www.linux-vserver.org/
[VxDLRef]	http://www.ens-lyon.fr/LIP/RESO/Software/vxdl/home.html

Glossary

AA	Authentication and Authorisation
AAA	Authentication, Authorisation and Accounting
AAI	Authentication and Authorisation Infrastructure
AM	Aggregate Manager
API	Application Program Interface
ASIC	[definition to be provided]
ASON	Automatically Switched Optical Network
BGP	Border Gateway Protocol
CCI	Connection Controller Interface
CIFS	Common Internet File System
CLI	Command Line Interface
CM	Component Manager
CMI	Customer Management Interface
CoMaR	Service Configuration and Activation
CPS	Circuit Processing System
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
DIOA	Distributed Interface Oriented Architecture
DWDM	Dense Wavelength-Division Multiplexed
EC2	Amazon Elastic Compute Cloud
ESB	Enterprise Service Bus
eTOM	TM Forum’s Enhanced Telecom Operations Map
e-VLBI	electronic Very Long Baseline Interferometry
EXPReS	Express Production Real-time e-VLBI Service
FAB	Fulfilment, Assurance, and Billing and Revenue Management
FCAPS	Fault, Configuration, Accounting, Performance, Security
FCS	Fast Circuit Switch
FI	Future Internet
FIRE	Future Internet Research and Experimentation
FOAM	FlowVisor OpenFlow Aggregate Manager
FPGA	Field Programmable Gate Array
G^{2}MPLS	Grid GMPLS
GEMBus	GÉANT Multi-domain Bus
GENI	Global Environment for Network Innovation
GENUS	GÉaNt virtUalisation Service

GGID	GENI Global Identifier
GLIF	Global Lambda Integrated Facility
GMC	GENI Management Core
GMPLS	Generalised Multi-Protocol Label Switching
GPO	GENI Project Office
GRE	Generic Routing Encapsulation
GSS-API	Generic Security Services API
GUI	Graphical User Interface
HDN	Health Data Network
HR	Human Resource
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IMF	Information Modelling Framework
InMaR	Service Information Management
InP	Infrastructure Provider
IP	Internet Protocol
IPNaaS	IP Network as a Service
iSCSI	Internet Small Computer System Interface
ITU	International Telecommunication Organisation
JIVE	Joint Institute for Very Long Baseline Interferometry in Europe
JRA1	GN3 Joint Research Activity 1, Future Network
JRA1 T4	JRA1 Task 4, Current and Potential Uses of Virtualisation
KVM	Kernel-based Virtual Machine
LHC	Large Hadron Collider
LICL	Logical Infrastructure Composition Layer
LR	Logical Resource
LXC	Linux Containers
MaR	Management Resource
MPLS	Multi-Protocol Label Switching
MRTG	Multi-Router Traffic Grapher
NaaS	Network as a Service
NAS	Network Attached Storage
NCP	Network Control Plane
NEXPReS	Novel Explorations Pushing Robust e-VLBI Services
NFS	Network File System
NGOSS	Next Generation Operations Support System
NIC	Network Interface Controller
NIPS	Network + IT Provisioning Service
NIPS UNI	Network + IT Provisioning Service User-Network Interface
NLI	NCP-LICL Interface
NLR	National Lambda Rail
NMS	Network Management System
NOVI	Networking innovations Over Virtualised Infrastructures
NREN	National Research and Education Network
NSF	National Science Foundation (US)
OE	Orchestration Engine

OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OFIAS	OpenFlow Switch In A Slice
Op-VNI	Optical Virtual Network Infrastructure
OS	Operating System
OSPF	Open Shortest Path First
OSS	Operations Support Systems
PaaS	Platform as a Service
PCE	Path Computation Element
PCI	Peripheral Component Interconnect
PCN	Programmable Core Nodes
PEC	Programmable Edge Clusters
PEN	Programmable Edge Nodes
PF	Programmable Framer
PIP	Physical Infrastructure Provider
PLC	PlanetLab Consortium
PoMaR	Service Policy Management
PoP	Point of Presence
PP	Packet Processor
PPS	Packet Processing System
PR	Physical Resource
PSNC	Poznań Supercomputing and Networking Centre
PTM	Panlab Testbed Manager
PWN	Programmable Wireless Nodes
QuMaR	Service Quality Management
R	Registry
R&E	Research and Education
RA	Resource Adapter
RAL	Resource Adaptation Layer
REST	Representational State Transfer
RIP	Routing Information Protocol
RP	Resource Provider
RSpec	Resource Specification
S3	Simple Storage Service
SaaS	Software as a Service
SC	Service Consumer
SCS	Service Consumer Stack
SDH	Synchronous Digital Hierarchy
SFA	Slice-based Federation Architecture
SLA	Service Level Agreement
SLI	SML to LICL Interface
SLS	Service Level Specification
SM	Slice Manager
SML	Service Middleware Layer
SMP	Symmetric Multiprocessing
SNIA	Storage Network Industry Association
SNMP	Simple Network Management Protocol

SOA	Service-Oriented Architecture
SSH	Secure Shell
SSM	Shared Storage Model
STREP	Specific Targeted Research Project
STS	Security Token Service
TM Forum	TeleManagement Forum
TrMaR	Service Trouble Management
UCLP	User-Controlled Lightpath Provisioning
UHDM	Ultra High Definition Media
UML	Unified Modelling Language
URI	Universal Resource Indicator
VCT	Virtual Customer Testbed
VI	Virtual Infrastructure
VIMS	Virtual Infrastructure Management System
VIO	Virtual Infrastructure Operator
VIP	Virtual Infrastructure Provider
VITM	Virtual IT Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNC	Virtual Network Controller
VNE	Virtual Network Embedding
VNet	Virtual Network
vNIC	Virtual Network Interface Card
VNO	Virtual Network Operator
VNP	Virtual Network Provider
VNS	Virtual Network Service
vOFS	Virtual OpenFlow Switches
VOSS	Virtualised Operations Support Service
VPC	Virtual Private Cloud
VPN	Virtual Private Network
VR	Virtual Resource
VRP	Virtual Resource Pool
vSMP	Virtual SMP
vSwitch	Virtual Switch
WDM	Wavelength-Division Multiplexing
WoR	Worker Resource
WS	Web Service
WSS	Wavelength Selective Switch
XML	Extensible Markup Language