|  |  |
| --- | --- |
| **G****FD-I.???**  **ISOD-RG**  **Version: 3**  **Late Updated: 14 May 2012** | **Editor(s)**  Chin Guok (ESnet)  **Contributors**  Scott Campbell (CRC)  Tangui Coulouarn (Forksningsnettet)  Yuri Demchenko (UvA)  Michal Giertych (PSNC)  Joan Antoni Garcia Espin (I2CAT)  Eduard Grasa (I2Cat)  Chin Guok (ESnet)  Radek Krzywania (PSNC)  Tomohiro Kudoh (AIST)  Mathieu Lemay (Inocybe Technologies)  Atsuko Takefusa (AIST)  Alexander Willner (Univ. of Bonn)  Yufeng Xin (RENCI)  TBA |

On-Demand Infrastructure Services Provisioning Best Practices

**Abstract**

The aim of this document is to provide an overview of best practices in on-demand provisioning of infrastructure services that includes both traditional Network Resources Provisioning Systems (NRPS) and emerging Cloud based infrastructure services. These provisioning processes must be both sufficiently explicit and flexible to dynamically instantiate complex task or project oriented infrastructures comprising of compute, storage, and application resources, as well network infrastructures interconnect them.

The proposed document summarises discussions among members of the OGF ISoD Research Group and aims to facilitate conversations on achieving interoperability, and effective use and development of modern and future infrastructure services provisioning systems.

Contents

1. Introduction 4

2. Infrastructure Services definition 5

2.1 General Infrastructure Definition 5

2.2 Infrastructure services definition in the context of this document 7

2.3 ISOD provisioning systems and general requirements 7

3. Network Resources Provisioning Systems (NRPS) 7

3.1 Argia 7

3.2 AutoBAHN 10

3.3 G-lambda and GridARS 14

3.4 OSCARS 16

3.5 Other NRPS 18

4. General and Cloud Oriented Network Infrastructure Services Provisioning 18

4.1 GENI-ORCA: A Networked Cloud Operating System for Extended Infrastructure-as-a-Service (IaaS) 18

4.1.1 ORCA architecture and its information model 19

4.1.2 NDL-OWL: Ontology-based cloud resource representation 20

4.1.3 IaaS service interface for infrastructure control 20

4.1.4 Cross-aggregate Stitching 21

4.2 GEYSERS Generalised Infrastructure Services Provisioning 21

4.2.1 GEYSERS Architecture 21

4.2.2 Physical Infrastructure 22

4.2.3 Logical Infrastructure Composition Layer (LICL) 23

4.2.4 Network + IT Control Plane (NCP+) 24

4.3 GEANT Services Virtualisation 26

4.4 HERO (Hybrid Networking Project for Research Oriented Infrastructure) 26

5. Provisioning infrastructure services in Clouds 26

5.1 Amazon Web Services 26

5.2 RackSpace 26

5.3 Google Application Engine 26

6. Existing standards 26

6.1 NIST Cloud Computing related standards 27

6.1.1 NIST Cloud Computing related activities and Standards 27

6.1.2 NIST Cloud Computing Reference Architecture (CCRA) 28

6.2 IEEE Intercloud Working Group (IEEE P2302) 31

6.3 IETF 32

6.4 ITU-T Focus Group Cloud Computing 33

6.5 Related activities at OGF 33

6.5.1 OCCI – Open Cloud Computing Interface 33

6.5.2 Network Service Interface Working Group (NSI-WG) 34

6.5.3 Network Markup Language Working Group (NML-WG) 34

7. Existing Cloud Middleware for Infrastructure Services Provisioning 34

7.1 OpenNebula 34

7.2 OpenStack 34

7.3 Eucalyptus 35

8. Projects on Infrastructure Services Provisioning – To be discussed 35

9. Proposed Taxonomy and Analysis 35

10. Summary and Recommendations 35

11. References 35

# Introduction

Dynamic provisioning and resource allocation has been a long-standing practise in many technology disciplines. Reserving cycles on a compute cluster or supercomputer, allocating space on a disk storage system, and carving out bandwidth in a network are common functions. However with the advent of Cloud Computing, co-scheduling and dynamic provisioning of resources across the various disciplines (i.e. compute, storage, applications, and networks) to create complex virtual infrastructures is revolutionary.

The growth Cloud Computing in recent years has advanced the development of technologies and methodologies in provisioning infrastructure. Cloud computing services can come in three forms, Infrastructure as a Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

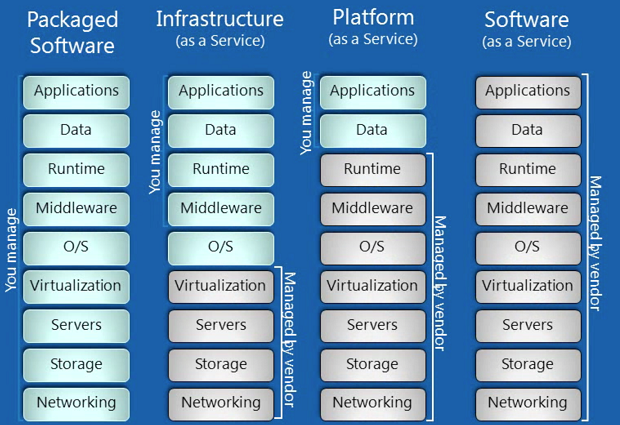


Figure 1.

1 IaaS, PaaS, SaaS definitions

IaaS is typically defined to encompass the physical data center infrastructure (e.g. building, racks, power, cooling, etc), network elements (e.g. cables, switches, routers, firewalls, etc), compute host/servers (e.g. CPU, GPU, memory), storage nodes/arrays (e.g. disks, caches, etc), and a virtualization layer in which combinations of compute and storage resources can be customized for each client. IaaS, sometimes referred to as “Hardware-as-a-Service”, in essence removes the requirement for an end-user or business to purchase and maintain physical hardware in order to develop software, execute a job, or provide a service. Examples of commercial IaaS offerings include Amazon Web Services [1], GoGrid [2] , OpenStack [3] , Rackspace [4] , and VMware [5] .

PaaS builds upon IaaS by adding an operating system, and infrastructure/middleware software (e.g. databases, runtime engines, etc). PaaS provides a “ready-to-go” framework environment for application management, design, and collaborative development. Examples of commercial PaaS offerings include Amazon Beanstalk [6], CloudFoundry [7], Google AppEngine [8], Microsoft Windows Azure [9], and SaleForce Force.com [10].

SaaS completes the Cloud module by including the user’s data and applications running on top of a PaaS. SaaS, also commonly referred to as “on-demand software”, includes most, if not all, common Cloud applications that are accessible through a web browser such as Apple iCloud, Netflix, Dropbox, and Google Gmail. Examples of commercial SaaS offerings include Cloud9 Analytics [11], CVM Solutions [12], GageIn [13], and KnowledgeTree [14].

The remainder of this document proposes to present an overview and taxonomy of infrastructure provisioning best and current practices in order to provide useful information for developing common and future infrastructure services provisioning models, architectures and frameworks.

# Infrastructure Services definition

## General Infrastructure Definition

Infrastructure is the basic physical and organizational structures needed for the operation of a society or enterprise, or the services and facilities necessary for an economy to function. Viewed functionally, infrastructure facilitates the production of goods and services; for example, roads enable the transport of raw materials to a factory, and also for the distribution of finished products to markets. In military parlance, the term refers to the buildings and permanent installations necessary for the support, redeployment, and operation of military forces.

Etymologically, the word “infrastructure” was first used in the English language in 1927 to define: “The installations that form the basis for any operation or system.” In this context there is a distinction between “hard” and “soft” infrastructures: “Hard” infrastructure includes transport, energy, water communication; “Soft” infrastructure includes institutional, industrial, social infrastructure and facilities.

The Internet infrastructure is both multilayered and multileveled, encompassing both “hard” and “soft” elements such as optical fibres, transponders, switches, and routers, as well as the protocols and other basic software necessary for the transmission of data.

To support the standardisation of open Information Technologies (IT), the Open Group [Ref??], in reference to their Integrated Information Infrastructure Reference Model (III-RM), proposes/defines the characteristics of infrastructure as follows:

* Infrastructure supports business processes
* Integrated information to prevent inconsistent and potentially conflicting pieces of information from being distributed throughout different systems
* Integrated access to information so that access to all necessary information is through one convenient interface

The following components are necessary in infrastructure operation:

* Applications and applications platform
* Operating System and Network services
* Communication infrastructure
* Infrastructure application including management tools

An alternate definition for generic infrastructure is proposed by Sjaak Laan [15], wherein he surmised that the most important aspects of IT infrastructure are:

* *“IT infrastructure consists of the equipment, systems, software, and services used in common across an organization, regardless of mission/program/project. IT Infrastructure also serves as the foundation upon which mission/program/project-specific systems and capabilities are built. (cio.gov - the website for the United States Chief Information Officers Council)”*
* “All of the components (Configuration Items) that are needed to deliver IT Services to customers. The IT Infrastructure consists of more than just hardware and software. (ITILv2)*”*
* “All of the hardware, software, networks, facilities, etc., that are required to Develop, Test, deliver, Monitor, Control or support IT Services. The term IT Infrastructure includes all of the Information Technology but not the associated people, Processes and documentation. (ITILv3)*”*
* “Information technology infrastructure underpins the distributed operational and administrative computing environment. Hidden from the application-based world of end-users, technology infrastructure encompasses the unseen realm of protocols, networks, and middleware that bind the computing enterprise together and facilitate efficient data flows. Yet information technology infrastructure involves more than just the mechanics of data systems; it also includes people providing support and services. (Technology Governance Board Definition of Information Technology Infrastructure)*”*
* ”Infrastructure is the shared and reliable services that provide the foundation for the enterprise IT portfolio. The implementation of an architecture includes the processors, software, databases, electronic links, and data centers as well as the standards that ensure the components work together, the skills for managing the operation etc. (Goethe University of Frankfurt, [http://www.is-frankfurt.de/](http://www.is-frankfurt.de/" \t "_blank))*”*

Sjaak further describes the typical characteristics of IT infrastructure as:

* *“IT infrastructure is usually shared by a multiple applications”*
* *“IT infrastructure is more static and permanent than the applications running upon it”*
* *“The management of the infrastructure is disconnected from the system management of the applications running on top of it”*
* *“The departments owning infrastructure components is different from the department owning the applications running on it”*

## Infrastructure services definition in the context of this document

In the context of Cloud based and general virtualized services, the infrastructure is defined as the total set of foundational components and non-functional attributes that enable applications to execute. Foundational infrastructure components include servers, operating systems, Virtual Machines (VMs), virtualization applications, (distributed) data-centers, network resources, and end-user devices. Non-functional infrastructure attributes include security, monitoring, management policies, and SLAs.

It is important to understand that Cloud infrastructures can be widely distributed over large geographical areas, which presents a critical requirement for networking resources to an in integral part of a Cloud’s internal infrastructure. In addition, networking is needed to interconnect Clouds together, and provide “last mile” access from the end-user. As such, provisioned infrastructure services must be characterized to include the following attributes/features:

* Topology definition for infrastructure services that encompass compute, storage, and network resources
* Infrastructure/topology description formats or schemas
* Related topology features or characteristics, and transformation operations (homomorphic, isomorphic, QoS, energy aware etc.)
* (TBD)

## ISOD provisioning systems and general requirements

TBD

# Network Resources Provisioning Systems (NRPS)

This section provides an overview of the best practices in network resource and services provisioning on demand. It presents several provisioning frameworks and systems varying from prototypes to production services, deployed on networks ranging from testbeds, and local-area networks, to wide-area backbones.

## Argia

Argia is an IaaS framework based product to create infrastructure-as-a-service solutions for optical networks. The main goal of Argia is to enable infrastructure providers to partition their physical networks/infrastructure and to give the control of the partitioned infrastructure to third parties (infrastructure integrators or APN administrators) for a period of time. These third parties may use the partitioned infrastructure in-house, or may deploy some intelligent software on top of the resources (like Chronos, the resource reservation service) to provide services for their end users, or they may even further partition the infrastructure and rent it to other users.

Argia is the evolution of the UCLP CE software; it is an ongoing effort towards creating a commercial product that can be deployed in production optical networks. Table 3.1.1 shows the network elements supported by the current release of Argia (Argia 1.4). Table 3.1.2 illustrates the networks and testbeds where Argia 1.4 has been deployed in the past or is still currently deployed, and what is being used for.

|  |  |  |
| --- | --- | --- |
| Vendor | Model | Technology |
| Cisco | ONS 15454 | SONET and SDH |
| Nortel | OME 6500 | SONET and SDH |
| Nortel | HDXc | SONET and SDH |
| Nortel | OPTera Metro 5200 | DWDM OADM |
| Calient | FiberConnect PXC | Photonic Cross Connect (PXC) |
| W-onesys | Proteus | DWDM ROADM |
| Cisco | Catalyst 3750, 6509 | Basic VLAN Management |
| Arista | 7124S | Basic VLAN Management |
| Allied Telesis | AT8000, AT9424 | Basic VLAN Management |
| Foundry | RX4 | Basic VLAN Management |

Table 3.1.1 Network elements supported by Argia 1.4

|  |  |
| --- | --- |
| Network or testbed | What is being used for |
| CANARIE network | Beta testing for use in production network  HPDMnet research project |
| STARlight (GLIF GOLE) | HPDMnet research project |
| PacificWAVE (GLIF GOLE) | HPDMnet research project |
| KRlight (GLIF GOLE) | PHOSPHORUS research project  HPDMnet research project |
| CRC Network | PHOSPHORUS research project |
| i2cat Network | PHOSPHORUS research project |
| University of Essex testbed | PHOSPHORUS research project |
| Poznan Supercomputing Center | PHOSPHORUS research project |
| DREAMS Project testbed | DREAMS research project |

Table 3.1.2 Current and past Argia deployment

Argia’s software architecture, depicted in Figure 3.1.1, is based on the IaaS Framework software. Argia’s software modules are the Optical Switch Web Services (WS) (a device controller service), the Connection WS and the Articulated Private Network (APN) Scenarios WS (End User Services).

argia arch..emf

Figure 3.1.1 Argia 1.4 Service Oriented Architecture

The Optical Switch WS is a Web Services Resource Framework (WSRF) based web service that can interact with one or more optical switch physical devices. The physical device state (inventory, including physical and logical interfaces, list of cross-connections, alarms and configuration) is exposed as a WS Resource, so that clients of the Optical Switch WS can access the state of the physical device by querying the resource properties. The Optical Switch WS interface provides a series of high level operations that encapsulate the physical device functionality.

Multi-vendor support is accomplished through the use of the IaaS Engine, a Java based framework to create drivers for physical devices. The Engine’s interface provides a Java-based model of the physical device’s state that satisfies two needs:

* Engine to Optical Switch WS communication: the engine fills the model attributes with the information of the physical device, allowing the Optical Switch WS to get the latest physical device information.
* Optical Switch WS to Engine communication: the Optical Switch WS fills some model attributes to request the Engine to perform some actions over the physical equipment; such as making a cross connection.

The Engine also provides abstractions to create the commands that the physical device understands, abstractions to group this commands into atomic actions, protocol parsers to generate the required command structure and transports to send and receive command through the network. The following example illustrates how the Engine’s APIs are applied to a particular use case: Let’s imagine that we want to create a driver that is capable of performing cross-connections on the Nortel OME 6500 network element. First of all, the driver developer would select the appropriate protocol parser, in this case TL-1 (the Engine allows to easily create new protocol parser implementations and new transports in case a particular protocol or transport is not already supported). Next, an adequate transport protocol is selected, for instance TCP. The next step would be to create the required commands to perform a cross connection: a command to login into the switch, another one to perform the cross-connection and another one to logout. Finally the developer would create the “Make Cross-Connection” action that grouped the three mentioned commands in a single atomic operation.

The Connection WS is also a WSRF web service that manages one or more connection resources (connections can be one to one, one to many, or loopback). Each connection resource has pointers to the set of network resources that are connected together. To create a connection, first the Connection WS classifies all the resources belonging to the same connection per optical switch; next it extracts the relevant parameters from the network resources (like the slots/ports/channels, the bandwidth, a cross-connection description), then it issues all the required “invoke” messages to the Optical switch WSs and finally it updates the state of the network resources.

Finally, the APN Scenarios WS is the evolution of the Custom APN Workflow. This service can setup and tear down preconfigured topologies consisting in a set of connections in an APN. To achieve its goal, when the “setup” operation is called on an APN Scenarios Resource, the APN Scenarios WS calls the Connection WS to create all the connections required by the scenario. Tearing down a scenario is a similar process: the Scenarios WS calls the “destroy” operation on each of the connection resources that have been created in the setup operation.

## AutoBAHN

The AutoBAHN [16] tool was conceived during the GÉANT2 project (2005-2009), and continues in the GÉANT3 project (2009-2013). The objective was to create a generic multi-domain system that was be able to integrate heterogeneous NRENs infrastructures and could be used in the tool-independent GÉANT Bandwidth on Demand (BoD) Service [ref: <http://bod.geant.net>]. From the very beginning, an emphasis was placed on scalability and flexibility to dynamically control a divers range of hardware. With the reduction of manpower to set up pan-European circuits, it was necessary for AutoBAHN to demonstrate security and reliability in order to be deployed in operational environments.

At the very beginning of the project, a long discussion on requirements and target environments were provided by multiple NREN representatives, resulting in a well thought out concept of the architecture. This concept allowed the creation of a very scalable distributed tool, which was the basis of the dynamic BoD initiative within the GÉANT network and associated NRENs. Each deployment (defined by distinct administrative domains) consists of the same building blocks, which form a three level hierarchy, each with its distinct responsibilities, as depicted in Figure 3.2.1.

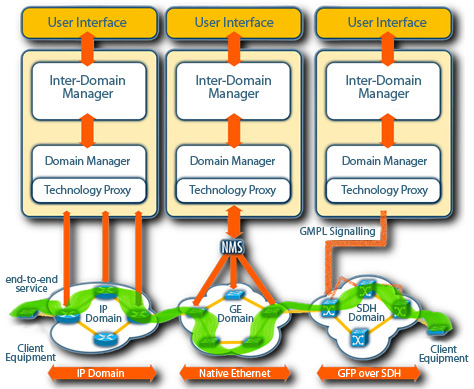


Figure 3.2.1 AutoBAHN architecture overview   
(figure from <http://www.geant.net/service/autobahn/User_Experience/Pages/UserExperience.aspx>)

The Inter-Domain Manager (IDM) at the top of the hierarchy is responsible for handling user requests and assuring inter-domain collaboration, including exchanging information with neighbouring AutoBAHN instances (IDMs in adjacent domains) or other provisioning systems. This module implementation is identical in all deployments, independently from the network environment and technology used. Its knowledge is restricted to an abstracted topology view of the network, which is technology agnostic and hides sensitive details (e.g. bandwidth utilisation), usually simplifying a domain’s view to a single cloud with interfaces to adjacent domains. The IDM is however aware of the global abstracted topology, which provides reachability and routing information for the entire service, and is used by the inter-domain Path Finder module to determine global circuits.

The IDM can contact a Domain Manager (DM) module, which is lower in the hierarchy, and is unaware of global connectivity. Instead it has all the details of local domain’s network technology, its hardware, intra-domain connections and time driven data or events. The DM is equipped with a set of tools that can verify, schedule or configure circuits within a single domain, assuring resources to be available on time and according to the user’s request. DMs are technology specific, requiring each implementation to be customized based on the local domain’s technology, administrative requirements, topology database used, etc. The responsibility of a DM is within the boundaries of a single administrative domain.

Finally, DMs use the Technology Proxy (TP) modules, which are at the bottom of the hierarchy, to communicate to the network hardware, or more likely local domain Network Management System (NMS). TPs are simple stateless proxies which translates generic DM messages into vendor specific commands in order to create and tear down circuits within the local domain. Domain administrators are encouraged to use existing tools (like NMS) to configure the network, as it simplifies the TP implementation, which is different for every domain and must respect local requirements, network features, configuration schemas, etc. While in most cases, AutoBAHN delivers out-of-the-box TPs for a range of supported technologies, it is possible to adapt the TP to use specific API, SNMP, or CLI based access to configure the hardware,

The Figure 3.2.2 presents a successful scenario of an end-to-end reservation workflow of a circuit traversing three independent domains – A, B, and C. The IDMs coordinates with other IDMs, which in turn communicates to the various DMs to verify resources and configure circuits within each domain along the reservation path. A global circuit is only delivered to the end user when all TPs have created the local segments of the end-to-end circuit within their respective domains.



Figure 3.2.2 AutoBAHN circuit request processing

The following outlines the reservation workflow as depicted in Figure 3.2.2. The User request is accepted by the IDM in domain A (1), which computes the intern-domain stitching points for the global reservation path (2). The IDM then contacts its local DM (in domain A) (3) in order to verify if there are sufficient resources to fulfil the request locally. The DM performs intra-domain path finding to investigate which exact domain resources will be used for the proposed circuit and compares that against a calendar of scheduled reservations to prevent resources overlapping. If sufficient resources are found, the DM reports back to the IDM (4), which in turn contacts the neighbouring IDM in domain B (5). The IDM in domain B performs the same local resource check of the local DM (6, 7, 8), and if successful, the same process is repeated in domain C (9, 10). A daisy chain model is used for communication in this workflow. The last IDM on the reservation path must take a decision on whether the circuit can be created end-to-end according to information collected from all the domains including itself. This information involves critical parameters that needs to be configured between domains or must be common for the entire global path, e.g. VLAN identifier for Ethernet circuits (if no VLAN translation is possible). If the decision was positive, the IDM in domain C orders the local DM to schedule the circuit and book resources in the calendar (11), and then sends a notification back to the neighbouring IDM on the reservation path (12). When all domains confirm resource availability and book the resources (13, 14, 15), the user is notified about the successful reservation (16). When the start time as specified by the user‘s service request arrives, each DMs initiates the local segment configuration using the TP modules to build the end-to-end circuit.

A critical function in the circuit reservation process is path finding. In AutoBAHN, it first estimates the global reservation path, which is then verified by local IDMs, resulting in a two stage process – inter- and intra-domain path finding. Since IDMs are unaware of exact resources available within particular domains, they need to be assisted by DMs to complete the whole process. In the event that a single domain cannot allocate resources for a particular circuit, an inter-domain path finding process can be repeated at the IDM which initiated the request to avoid such domains. The process can be iterated until resources are found in all the required domains, or no global path can be defined.

AutoBAHN by design involves multiple attributes that can be used to specify a circuit request. The main attributes are obligatory and include the reservation start and end time, circuit end points, and capacity. The start and end time can be specified in the granularity of minutes, and the circuit instantiation function is designed to take into account the necessary setup time, assuring the circuit is available to the end users when needed. End points can be selected using a web page based GUI that is easy to navigate and select. Finally, the capacity attribute is defined by the user in bits per second (bps), which includes both the user’s payload and any link otherhead. A user can also optionally specify which VLAN identifiers to be used at the start and end points of the circuit. In addition, users may select the maximum delay (in ms) and required MTU for the path, in cases where this attributes matters, in addition to the circuit capacity. Finally a user can influence the path computation by specifying a “white” (wanted) or “black” (unwanted) list of abstract topology links to be used by circuit. This is particularly useful to explicitly require paths to pass through particular domains or use specific interfaces where alternatives are possible.

Since the beginning of the GÉANT3 project in 2009, AutoBAHN has been deployed within GÉANT and several associated NRENs to enable a BoD service. During the GÉANT BoD service pilot, the AutoBAHN tool was deployed in half of the 8 NRENs (GRNet, HEAnet, PIONIER, Forksningsnettet, Nordunet, Carnet, Surfnet, and JANET), as well as DANTE, which manages the GÉANT backbone (see Figure 3.2.3). The BoD service was rigorously verified in this environment and is now offered as a production service to users. The AutoBAHN tool supports several technologies including Ethernet, Carrier Grade Ethernet, MPLS, and SDH, and can interact with a variety of equipment vendors such as Juniper, Cisco, Brocade, and Alcatel. The flexibility in its modular architecture provides an opportunity for AutoBAHN to be adapted by any infrastructure regardless of local technologies, administration procedures, and policies.

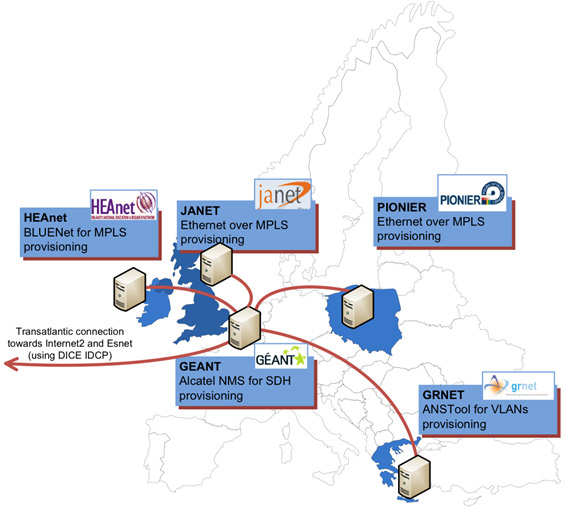


Figure 3.2.3 AutoBAHN pilot deployment

(figure from <http://www.geant.net/service/autobahn/User_Experience/Pages/UserExperience.aspx>)

The AutoBAHN tool is evolving to include enhanced functionality such as; advanced user authorisation, adding new technologies and vendors support, accounting mechanisms, and resiliency. In addition, AutoBHAN is integrating the OGF NSI CS protocol to promote interoperability with other provisioning systems. The development process consists of making existing tools more robust and stable, while concurrent research is aimed at investigating new requirements or functionalities that can increase the value of the service for the end users.

## G-lambda and GridARS

The G-lambda [16] project, started in 2004, is a collaboration between Japan’s industrial and governmental laboratories, KDDI R&D Laboratories, NTT, NICT and AIST. The goal of this project is to define a Web service-based network service interface, named GNS-WSI (Grid Network Service - Web Service Interface), through which users or applications can request end-to-end bandwidth-guaranteed connection.

While GNS-WSI v.1 and v.2 were defined as an interface for network services, GNS-WSI v.3 (GNS-WSI3) has been defined as an interface for heterogeneous resource services, which enables requests and coordination of heterogeneous resources (e.g. computers, networks, and storage) uniformly. GNS-WSI3 has been designed as a polling-based two-phase commit protocol, which enables distributed transactions.



Figure 3.3.1 Reference model of GNS-WSI3

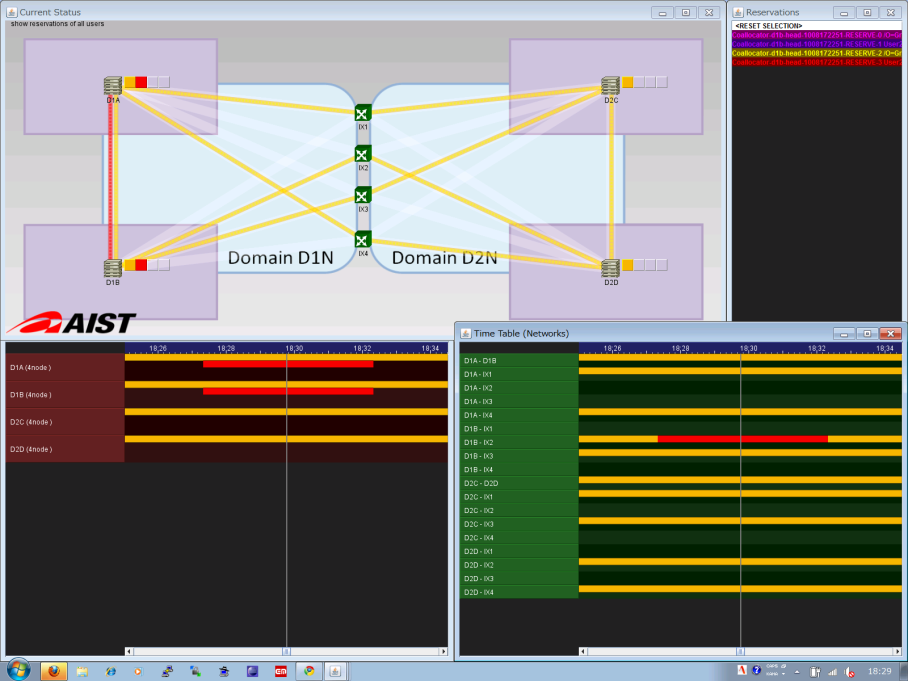
Figure 3.3.1 shows the reference model of GNS-WSI3. This model consists of a Global Resource Coordinator (GRC), which coordinates heterogeneous resources via Resource Managers (RMs), which manage local resources directly. The NRM, CRM, and SRM in Figure 3.3.1 denote RMs for networks, computers and storage, respectively. GRCs and RMs work together to provide users virtualized resources. GRCs can be configured in a coordinated hierarchical manner, or in parallel, where several GRCs compete for resources with each other on behalf of their requesters.

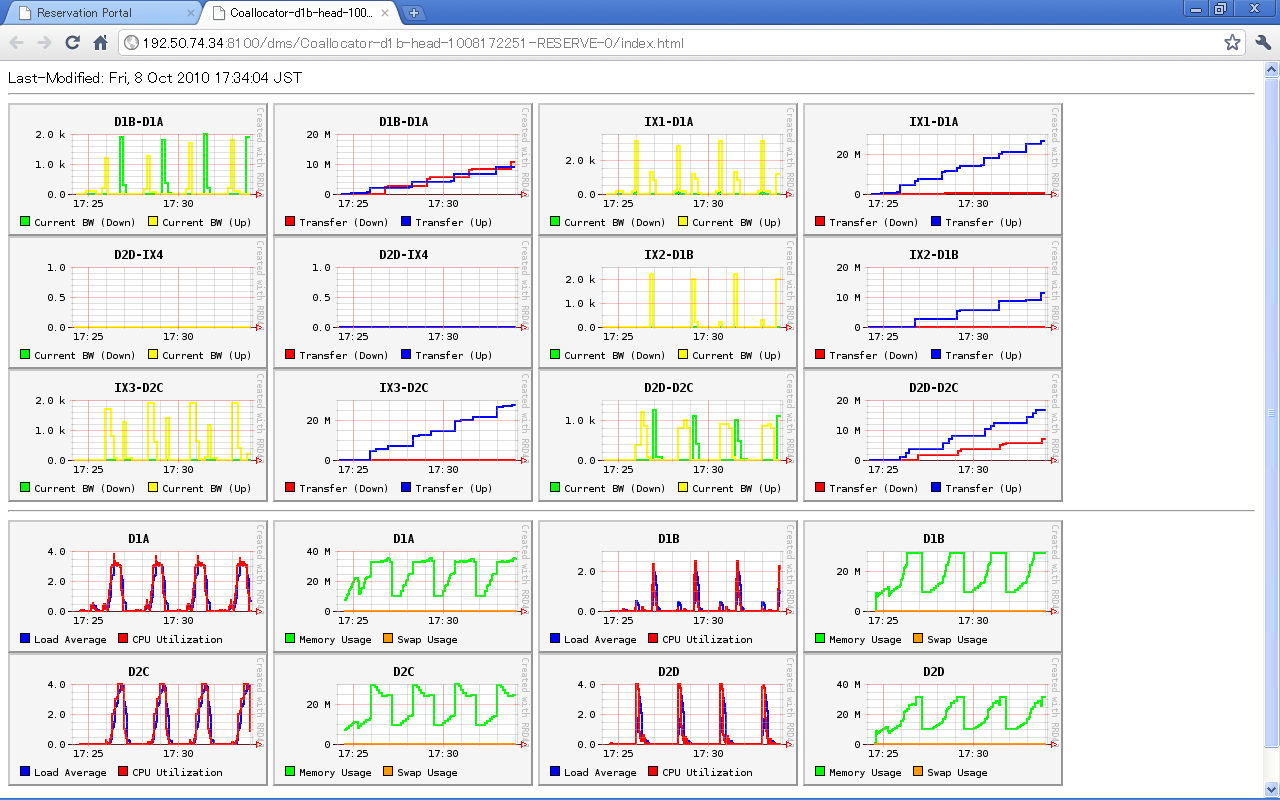
GNS-WSI3 provides SOAP-based operations to reserve, modify, release various resources and query available resources, as shown in Table 3.3.1. RsvID and CmdID indicate IDs of the requested reservation and each command, such as reserve, modify or release. Each command behaves as pre-procedure instruction, with the commit or abort operations to execute or abort the command. Users can confirm reservation or command status via getResourceProperty and obtain information of available resources, provided by each RM, via getAvailableResources. A requester sends create, reserve, getResourceProperty(CommandStatus) and commit operations to resource coordinators or providers in a normal reservation process.

|  |  |  |
| --- | --- | --- |
| Operation | Function | Input / Output |
| create | Initialize | - / RsvID |
| reserve | Make resource reservation | RsvID, requirements on resources and time / CmdID |
| modify / modifyAll | Modify part / all of reserved resources | RsvID, requirements on resources and time / CmdID |
| release / releaseAll | Release part / all of reserved resources | RsvID, resource ID / CmdID |
| commit / abort | Execute / abort the command | CmdID / - |
| getResourceProperty | Return the property values (E.g., Reservation / Command status) | Property names / Property values |
| getAvailableResources | Provide available resource information | Conditions / Available resource information |

Table 3.3.1 GNS-WSI3 operations

AIST has been developing a GNS-WSI reference implementation, called the GridARS[2] resource management framework, which provides not only a resource management service, based on GNS-WSI, but also planning, provisioning and monitoring services. GridARS enables the construction of a virtual infrastructure over various inter-cloud resources and provides the requester its monitoring information, dynamically.





Reservation status

CPU utilization and load average

Network traffic and bandwidth

Figure 3.3.2 GridARS monitoring results. Reserved resource status (left) and   
network and computer resource status of a virtual infrastructure (right).

KDDI R&D Laboratories and NTT have also developed their own reference implementations of GNS-WSI. This was demonstrated at iGrid2005[3] as a single domain by using GNS-WSI v.1 and international multiple domains at GLIF2006[4] and GLIF2007 by using v.2. G-lambda was also involved in the Fenius[5] and OGF NSI[6] interoperation demo in 2010 and 2011 using GNS-WSI v.3.

[1] G-lambda, http://www.g-lambda.net/.

[2] GridARS, http://www.g-lambda.net/gridars/.

[3] A. Takefusa, M. Hayashi, N. Nagatsu et al., "G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS", Future Generation Computing Systems, vol. 22, pp. 868-875, 2006.

[4] S. R. Thorpe and L. Battestilli, G. Karmous-Edwards et al., "G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US", Proc. GridNets2007, 2007.

[5] Fenius, http://code.google.com/p/fenius/.

[6] OGF (Open Grid Forum) NSI-WG (Network Service Interface Working Group), http://forge.gridforum.org/sf/projects/nsi-wg.

## OSCARS

The On-demand Secure Circuits and Advance Reservation System (OSCARS) was motivated by a 2002 U.S. Dept. of Energy (DOE) Office of Science High-Performance Network Planning Workshop that identified bandwidth-on-demand as the most important new network service that could facilitate:

* Massive data transfers for collaborative analysis of experiment data
* Real-time data analysis for remote instruments
* Control channels for remote instruments
* Deadline scheduling for data transfers
* “Smooth” interconnection for complex Grid workflows

In Aug 2004, DOE funded the OSCARS project to develop dynamic circuit capabilities for the Energy Sciences Network (ESnet). The core requirements in the design of OSCARS were based on the need for dynamic circuits to be:

* Configurable: The circuits are dynamic and driven by user requirements (e.g. termination end-points, required bandwidth, sometimes routing, etc.).
* Schedulable: Premium services such as guaranteed bandwidth will be a scarce resource that is not always freely available and therefore is obtained through a resource allocation process that is schedulable.
* Predictable: The service provides circuits with predictable properties (e.g. bandwidth, duration, reliability) that the user can leverage.
* Reliable: Resiliency strategies (e.g. re-routes) that can be made largely transparent to the user should be possible.
* Informative: The service must provide useful information about reserved resources and circuit status to enable the user to make intelligent decisions.
* Geographically comprehensive: OSCARS must interoperate with different implementations of virtual circuit services in other network domains to be able to connect collaborators, data, and instruments worldwide.
* Secure: Strong authentication of the requesting user is needed to ensure that both ends of the circuit are connected to the intended termination points; the circuit must be managed by the highly secure environment of the production network control plane in order to ensure that the circuit cannot be “hijacked” by a third party while in use.

In the latest release of OSCARS (v0.6), each functional component was implemented as a distinct “stand-alone” module with well-defined web-services interfaces. This framework permits “plug-and-play” capabilities to customize OSCARS for specific deployments needs (e.g. different AuthN/AuthZ models), or for research efforts (e.g. path computation algorithms). A description of each of the functional models is listed and discussed below.

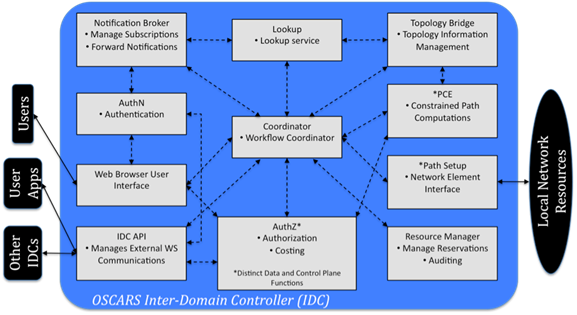


Figure 3.4.1 OSCARS software architecture

* Notification Broker: The Notification Broker is the conduit for notifying subscribers of events of interest. It provides users the ability to (un)subscribe to “topics”, which are then used as filters to match events. If an event matches a topic, the notification broker will send a notify message (as defined by WS-Notification [ref]) to the subscriber. In addition to circuit service users, the notification broker is used to notify the perfSONAR [ref] circuit monitoring service when a circuit is setup or torndown.
* Lookup: This module is responsible for publishing the location of the local domain’s externally facing services, as well as locating the service manager (e.g. Inter-Domain Controller/Manager) of a particular domain. The Lookup module currently utilizes the perfSONAR Lookup Service in order to perform its tasks.
* Topology Bridge: The Topology Bridge is responsible for fetching all the necessary topologies for the path computation to determine an end-to-end circuit solution. The Topology Bridge currently utilizes the perfSONAR Topology Service to pull down topologies.
* AuthN: This module is responsible for taking a validated identity (ID) token and returning attributes of the user. In ESnet, ID tokens can either be an x.509 DistinguishedName (DN), or a registered web interface user/password login, and attributes returned using SAML2.0 AttributeTypes.
* AuthZ: This module is the policy decision point for all service requests to OSCARS and manages permissions, actions, and attributes. It takes a list of user attributes, a resource, and a requested action and returns an authorization decision.
* Coordinator: The Coordinator is responsible for handling both client and inter-domain messages, and enforces the workflow of the service request.
* PCE: The Path Computation Engine (PCE) is responsible for finding the end-to-end path of a circuit. In OSCARS v0.6, the PCE is a framework that can represent a single monolithic path computation function, or a tree of smaller atomic path computation functions (e.g. bandwidth, VLAN, hop count, latency computations).
* Resource Manager: The Resource Manager is responsible for keeping the current state of the reservation request. It stores both the user’s original reservation requests as well as the solution path that was computed by the PCE.
* Path Setup: This module interfaces directly with the network elements and functions as the mediation layer between OSCARS and the data transport layer. The Path Setup module contains the vendor (e.g. Alcatel, Ciena, CISCO, Infinera, Juniper, etc) and technology (e.g. MPLS, GMPLS, Ethernet Bridging) specific details to instantiate the necessary circuits in the data plane.
* IDC API: The Inter-Domain Controller (IDC) API is responsible for external communications to client application/middleware and other IDCs. The IDC API currently supports the base as well as recent extensions to the IDCP v1.1 [ref] protocol.
* Web Browser User Interface (WBUI): The WBUI is the web user interface for users to create, cancel, modify, and query reservations, in addition to account management by administrators.

Since 2007, OSCARS has been supporting Ethernet Virtual Private Line (EVPL) production services in ESnet and is used to carrying about ½ of ESnet’s total traffic today. As of 2011, OSCARS has been adopted by over 20 networks worldwide, including wide-area backbones, regional networks, exchange points, local-area networks, and testbeds. In 2012, the installation base of the OSCARS software is expected to reach over 50 networks.

OSCARS is still evolving and expanding its feature sets and functionality to include capabilities such as protection services, multi-layer provisioning, anycast/manycast/multicast path computation, and the adoption of the OGF NSI CS protocol. These efforts have been made possible primarily due to an international collaboration of researchers, software developers, and network operators on the OSCARS project.

## Other NRPS

TBD

# General and Cloud Oriented Network Infrastructure Services Provisioning

## GENI-ORCA: A Networked Cloud Operating System for Extended Infrastructure-as-a-Service (IaaS)

ORCA is one of the GENI Control Frameworks and is being developed jointly with Duke University by the Networking Group at RENCI/UNC-CH [1, 4]. Based on the extended Infrastructure-as-a-Service (IaaS) cloud model, it can be regarded as an operating system for orchestrated provisioning of heteroge­neous resources across multiple federated substrate sites and domains. Each site is either a private IaaS cloud that can instantiate and manage virtual (eg. Eucalyptus or OpenStack) and physical machines, or a transit network domain that can provision bandwidth-guaranteed virtual network channels between its border interfaces (e.g., ESnet, NLR, or Internet2).

From the service provisioning perspective, ORCA can support multiple types of on-demand virtual infrastructure requests from users. We ﬁrst classify these requests as bound or unbound. The bound requests explicitly specify the sites for provisioning the virtual or physical hosts. For bound requests, the system determines the transit network providers needed to interconnect the components provisioned in the diﬀerent clouds via a constrained pathﬁnding algorithm. The unbound requests describe the virtual topology with required node and edge resources without specifying which site or sites the embedding should occur in. For unbound requests, the system selects a partitioning of the topology that yields a cost-eﬀective embedding of the topology across multiple cloud providers. Examples of typical requests include: (1) provisioning a group of hosts from a cloud; (2) provisioning a virtual cluster of VMs in a cloud, connected by a VLAN; (3) provisioning a virtual topology within a cloud; (4) provisioning an inter-cloud connection between two virtual clusters in two diﬀerent clouds; (5) Provisioning a virtualized network topology over multiple clouds. In each of the examples a request may be bound (partially or fully) or unbound.

### ORCA architecture and its information model

ORCA is a fully distributed system. An ORCA deployment consists of three types of components (called actors in ORCA): slice manager or SM (facilitating user’s topology requests), an aggregate manager or AM (one for each substrate provider) and a broker that encapsulates a coordinated allocation and autho­rization policy. Compared to the GENI architecture, ORCA has two major unique capabilities : (1) ORCA broker can actually provide resource broker­age service via policy enforcement, which includes coordinated allocation across multiple sites and substrate stitching [4]; (2) The three components all support pluggable resource management and access policies for accessing diﬀerent types of substrates and interfacing with diﬀerent user APIs. Internally ORCA actors use a well-deﬁned API (implemented using SOAP) that uses tickets and leases (signed promises of resources) annotated with property lists describing exact attributes of various resources. Tickets are the fundamental mechanism that allows Orca brokers to create complex policies for coordinated allocation and management of resources.

Resources within ORCA have a lifecycle and there are 5 types of models within this lifecycle. ORCA actors generate, update, and pass these models around to acquire resources (slivers) from multiple substrate aggregates, stitch them into an end-to-end slice upon users’ requests and pass control over the resources to the user. This interaction and information ﬂow life cycle is depicted in Figure 4.1.1.1.

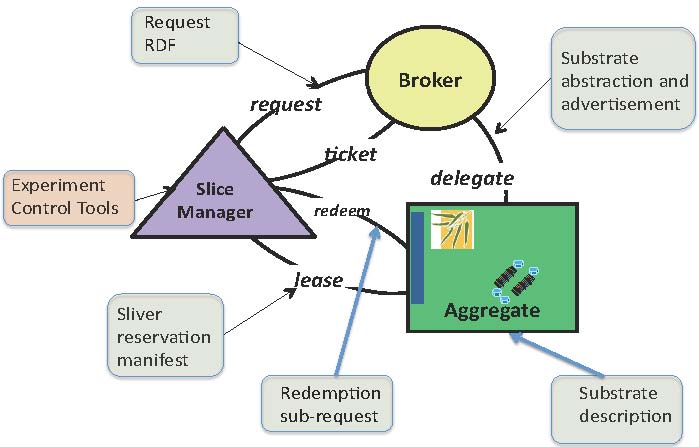


Figure 4.1.1.1 Information flow life cycle in ORCA architecture

**Substrate delegation model**

This is the abstract model to advertise an aggregate’s resources and services externally. ORCA AMs use this representa­tion to delegate advertised resources to ORCA broker(s). AMs may also use this representation to describe resources in response to queries or to advertise resources to a GENI clearinghouse. This model allows multiple abstraction lev­els, as diﬀerent AMs may want to expose diﬀerent levels of detail in resource and topology descriptions of their substrate.

**Slice request model**

This is the abstract model to represent user resource requests. A typical request might be a virtual topology with speciﬁc resources at the edges, generated by some experiment control tool. Our implementation allows the submission of a request via a GUI, automatic generation of a request from a point-and-click interface or the use of an XMLRPC API for submission and monitoring of the state of the request.

**Slice reservation model**

This is the abstract model used by ORCA bro­kers to return resource tickets to the SM controller. Each ticket contains a information on one or more slivers (individually programmable elements of a slice) allocated from a speciﬁc AM named in the ticket. The SM controller obtains the slivers by redeeming these tickets with individual resource provider AM actors. This model describes the interdependency relationships among the slivers so that the SM controller can drive stitching information for each sliver into the slice. Currently, ORCA uses the substrate delegation model, i.e. the ticket contains a resource type and unit count for the resources promised in the ticket, together with the complete original advertisement from the AM.

**Slice manifest model**

This is the abstract model that describes the topol­ogy, access method, state, and other post-conﬁguration information of the slivers within the requested slice. The manifest may be used as input by an intelli­gent tool to drive the user experiment. It also serves as a debugging aide as it presents detailed information about slice topology that includes details regard­ing resources acquired from intermediate resource providers.

### NDL-OWL: Ontology-based cloud resource represen­tation

ORCA uses a set of uniﬁed semantic schemas (ontologies) for representing the data models to describe resources from heterogeneous substrates. We developed NDL-OWL -an extension of the Network Description Language (NDL) [5] orig­inally developed to describe multi-layer transport network resources. In NDL­OWL we have developed ontologies to represent the the compute resources in cloud provider sites. As described above, we also developed the ontologies for the request, domain delegation, and manifest information models as well.

We use a number of mature semantic web software tools, like Protege [3], to create and maintain the ontologies. The Jena RDF/OWL Java package for Java [2], representing the semantic web in an internal graph structure, gives us a ﬂexible semantic query-based programming approach to implementing the policies for resource allocation, path computation, and topology embedding. It enables these functions by generically coding and operating on declarative speciﬁcations, rather than hard-coding assumptions about the resources into the policy logic.

### IaaS service interface for infrastructure control

Standard services and APIs with standard back-end infrastructure control ser­vices oﬀers a path to bring independent resource providers into the federation. We developed drivers for ORCA to call these APIs from popular cloud and network provisioning platforms, which include Eucalyptus, OpenStack, ESNet OSCARS, and NLR Sherpa.

To make topology embedding in clouds possible we also developed NEuca [1] a Eucalyptus (and now OpenStack) extension that allows guest VM conﬁgura­tions to enable virtual topology embedding within a cloud site. NEuca (pro­nounced nyoo-kah) consists of a set of patches and additional guest conﬁguration scripts installed onto the virtual appliance image, that enhance the functionality of a private Eucalyptus or OpenStack cloud without interfering with its normal operations. It allows VMs instantiated via Eucalyptus or OpenStack to have additional network interfaces, not controlled by Eucalyptus that are tied into speciﬁc VLANs or physical worker interfaces.

We also developed an ImageProxy service for uniform management of images across multiple cloud sites. It is a stand-alone caching server at each cloud site that enables the site to import images on demand from an Internet server. Thus a user is relieved from having to explicitly register an image with a speciﬁc cloud service. Instead the user speciﬁes the image location as a URL (and an image SHA-1 sum for veriﬁcation) and ImageProxy under ORCA control downloads and registers the image at the site where user slivers will be instantiated.

### Cross-aggregate Stitching

ORCA provides a general facility for cross-aggregate stitching that applies for network stitching and other stitching use cases as well. This feature enables ORCA to orchestrate end-to-end stitching across multiple aggregates. It can incorporate inter-domain circuit services oﬀered by third parties. Currently, ORCA uses dynamic layer-2 vlan circuit as the primary mechanism for end­to-end virtual networking. The stitching process uses explicit dependency and capability tracking within each inter-domain path in a slice to instantiate the resources in the proper order of their dependence on each other (e.g. one site may depend on the other to provide a circuit/VLAN tag before it can be stitched into a path).

Based on the switching and label swapping capability, the ORCA controller constructs a directed dependency DAG as it plans a slice’s virtual topology and the mapping to aggregates. The controller then traverses the DAG, instantiating slivers and propagating labels to their dependent successors as the labels become available.

References Cited

[1] GENI ORCA Control Framework. http://geni-orca.renci.org.

[2] Jena – A Semantic Web Framework for Java.

[3] The Prot´eg´e Ontology Editor and Knowledge Acquisition System.

[4] I. Baldine, Y. Xin, A. Mandal, C. Heermann, J. Chase, V. Marupadi,

A. Yumerefendi, and D. Irwin. Autonomic Cloud Network Orchestration: A GENI Perspective. In 2nd International Workshop on Management of Emerging Networks and Services (IEEE MENS ’10) , in conjunction with GLOBECOM’10, Dec. 2010.

[5] J. Ham, F. Dijkstra, P. Grosso, R. Pol, A. Toonk, and C. Laat. A distributed topology information system for optical networks based on the semantic web. Journal of Optical Switching and Networking, 5(2-3), June 2008.

## GEYSERS Generalised Infrastructure Services Provisioning

### GEYSERS Architecture

GEYSERS introduces a new architecture that re-qualifies the interworking of legacy planes by means of a virtual infrastructure representation layer for network and IT resources and its advanced resource provisioning mechanisms. 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 network and IT converged services. On the one hand, the service-oriented paradigm and IaaS framework enable flexibility of infrastructure provisioning in terms of configuration, accessibility and availability for the user. On the other hand, the layer-based structure of the architecture enables separation of functional aspects of each of the entities involved in the converged service provisioning, from the service consumer to the physical ICT infrastructure.

Figure 4.2.1.1 shows the layering structure of the GEYSERS architecture reference model. Each layer is responsible to implement different functionalities covering the full end-to-end service delivery from the service layer to the physical substrate. Central to the GEYSERS architecture and focus of the project are the enhanced Network Control Plane (NCP), and the novel Logical Infrastructure Composition Layer (LICL). The Service Middleware Layer (SML) represents existing solutions for service management and at the lowest level there is the Physical Infrastructure layer that comprises optical network and IT resources from different Physical Infrastructure Providers. Each of these layers is further described below.



Figure 4.2.1.1 GEYSERS layered architecture

### Physical Infrastructure

The GEYSERS physical infrastructure is composed of optical network and IT resources. These resources may be owned by one or more physical infrastructure providers and can be virtualized by the LICL. The term infrastructure refers to all physical network resources (optical devices/physical links) used to provide connectivity across different geographical locations and the IT equipment providing storage space and/or computational power to the service consumer. From the network point of view GEYSERS will rely on the L1 optical network infrastructure. The GEYSERS architecture is expected to be generic enough to cover most of the technologies used in the existing optical backbone infrastructures offered by today’s infrastructure providers/operators. Nevertheless, focus will be on Fiber Switch Capable (FSC) and Lambda Switch Capable (LSC) devices. From an IT point of view, IT resources are considered as service end-points to be connected to the edge of the network. IT resources are referred to physical IT infrastructures of IT such as computing and data repositories.

The physical infrastructure should provide interfaces to the equipment to allow its operation and management, including support for virtualization (when available), configuration and monitoring. Depending on the virtualization capabilities of the actual physical infrastructure, physical infrastructure providers may implement different mechanisms for the creation of a virtual infrastructure. In terms of optical network virtualization, GEYSERS considers optical node and optical link virtualization. Moreover, the virtualization methods include partitioning and aggregation.

* Optical node partitioning: It entails dividing an optical node into several independent virtual nodes with independent control interfaces by means of Software and Node OS guaranteeing isolation and stability.
* Optical node aggregation: It entails presenting an optical domain or several interconnected optical nodes (and the associated optical links) as one unified virtual optical switching node with a single/unified control interface by means of Software and Control/Signalling Protocols. The controller of the aggregated virtual node should manage the connections between the internal physical nodes and show the virtual node as a single entity.
* Optical link partitioning: It entails dividing an optical channel into smaller units. Optical fibres can be divided into wavelengths and wavelengths into sub-wavelength bandwidth portions that can be performed e.g. using advanced modulation techniques. The latter is a very challenging process especially when the data rate per wavelength is >100Gbps.
* Optical link aggregation: Several optical wavelengths can be aggregated into a super-wavelength with aggregated bandwidth ranging from wavelength-band, to fibre or even multi-fibre level.

After partitioning and aggregation, optical virtual nodes and links are included in a virtual resource pool used by the LICL to construct virtual infrastructures; thus, multiple virtual infrastructures can share the resources in the optical network. This means that isolation between the partitioned virtual resources has to be guaranteed at both data (physical isolation) and control level.

### Logical Infrastructure Composition Layer (LICL)

The LICL is a key component in the GEYSERS architecture. It is located between the physical infrastructure and the upper layers, NCP and SML. The LICL is responsible for the creation and maintenance of virtual resources as well as virtual infrastructures. In the context of GEYSERS, infrastructure virtualisation is the creation of a virtual representation of a physical resource (e.g., optical network node or computing device), based on an abstract model that is often achieved by partitioning or aggregation. A virtual infrastructure is a set of virtual resources interconnected together that share a common administrative framework. Within a virtual infrastructure, virtual connectivity (virtual link) is defined as a connection between one port of a virtual network element to a port of another virtual network element.

The LICL utilizes a semantic resource description and information modelling mechanism for hiding the technological details of the underlying physical infrastructure layer from infrastructure operators. Consequently, the LICL acts as a middleware on top of the physical resources and offers a set of tools that enable IT and Optical Network resource abstraction and virtualization. Moreover, the LICL allows the creation of virtual infrastructures using the virtualized resources and a dynamic on-demand re-planning of the virtual infrastructure composition. The LICL manages the virtual resource pool where virtual resources are represented seamlessly and in an abstract fashion using a standard set of attributes, which allows the enhanced Control Plane to overcome device dependency and technology segmentation. The LICL also brings the innovation at the infrastructure level by partitioning the optical and IT resources belonging to one or multiple domains. Finally, LICL supports the dynamic and consistent monitoring of the physical layer and the association of the right security and access control policies. LICL mainly supports the following functionalities:

* Physical resource virtualization
* Semantic resource description and resource information modelling
* Physical/virtual resource synchronization and monitoring
* Virtual infrastructure composition and management
* Virtual infrastructure planning/re-planning
* Security handling

The LICL requires privileged access to the physical infrastructure resources in order to implement isolation in an efficient manner. It also works as a middleware that forwards requests and operations from the NCP to the physical infrastructure native controllers. This is achieved by using a Virtual Infrastructure Management System (VIMS) that is a set of tools and mechanisms for control and management of its resources (Figure 5).



Figure 4.2.3.1: LICL basic functional decomposition

### Network + IT Control Plane (NCP+)

The GEYSERS network and IT control plane (NCP+) operates over a virtual infrastructure, composed of virtual optical network and IT resources, located at the network edges. The virtual infrastructure is accessed and controlled through a set of interfaces provided by the LICL for operation and re-planning services. The NCP+ offers a set of functionalities towards the SML, in support of on-demand and coupled provisioning of the IT resources and the transport network connectivity associated to IT services.

The combined Network and IT Provisioning Service (NIPS) requires the cooperation between SML and NCP+ during the entire lifecycle of an IT service. This interaction is performed through a service-to-network interface, called NIPS UNI. Over the NIPS UNI, the NCP+ offers functionalities for setup, modification and tear-down of enhanced transport network services (optionally combined with advance reservations), monitoring and cross-layer recovery.

The GEYSERS architecture supports several models for the combined control of network and IT resources. The NCP+ can assist the SML in the selection of the IT resources providing network quotations for alternative pairs of IT end points (assisted unicast connections). Alternatively the NCP+ can select autonomously the best source and destination from a set of end points, explicitly declared by the SML and equivalent from an IT perspective (restricted anycast connections). In the most advanced scenario, the NCP+ is also able to localize several candidate IT resources based on the service description provided by the SML, and computes the most efficient end-to-end path including the selection of the IT end-points at the edges (full anycast connections). This is a key point for the optimization of the overall infrastructure utilization, also in terms of energy efficiency, since the IT and network resources configuration is globally coordinated at the NCP+ layer.

The NCP+ is based on the ASON/GMPLS [3] and PCE [4] architectures, is enhanced with routing and signalling protocols extensions and constraints based route computation algorithms designed to support the NIPS and, on the other hand, to optimize the energy efficiency for the global service provisioning. Particularly the NCP layer implements mechanisms for advertisement of the energy consumption of network and IT elements as well as computation algorithms which are able to combine both network and IT parameters with energy consumption information to select the most suitable resources and find an end-to-end path consuming the minimum total energy. Figure 6 shows a high-level representation of the NCP+: the routing algorithms at the PCE operate over a topological graph created combining network and IT parameters with “green” parameters, retrieved from the SML (IT side) and the LICL (network side).

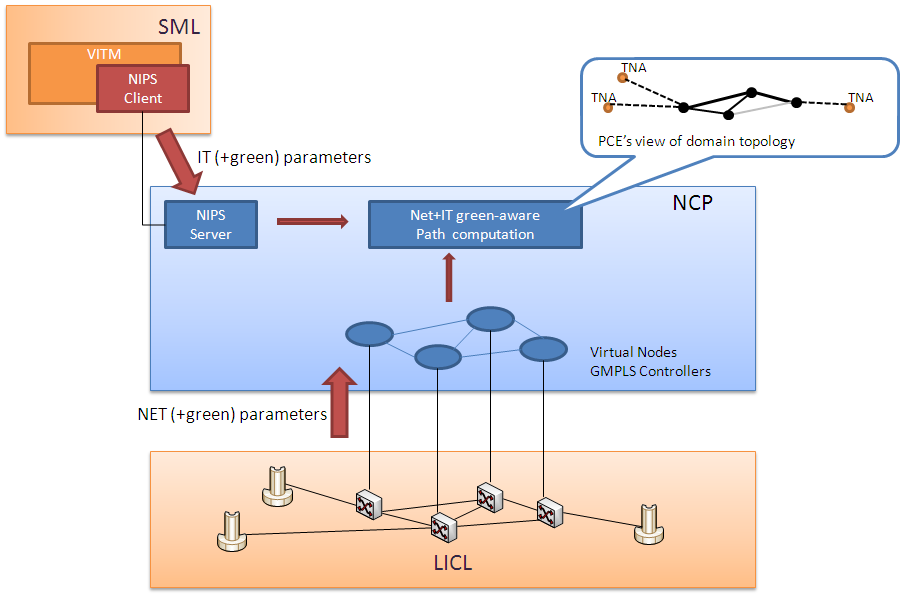


Figure 4.2.4.1: NCP – Network and IT energy-aware service provisioning

Finally, another key element for NCP+ is the interaction with the LICL in order to trigger the procedures for the virtual infrastructure dynamic re-planning on the network side. In case of inefficiency of the underlying infrastructure, the NCP+ requests the upgrade or downgrade of the virtual resources in order to automatically optimize the size of the virtual infrastructure. The involved algorithms take into account current network traffic, forecasts for resource availability and utilization in the medium and long terms, as well as specific SLAs established between provider and operator for dynamic modifications of the rented virtual resources.

[3] ITU-T G.8080/Y.1304 Recommendations, “Architecture for the Automatic Switched Optical Network (ASON)”, 2001

[4] A. Farrel, J.P. Vasseur, J. Ash, “A Path Computation Element (PCE) – Based Architecture”, IETF RFC 4655, August 2006

## OpenNaaS (GEANT Virtualisation Services)

TBA

## HERO (Hybrid Networking Project for Research Oriented Infrastructure)

# Provisioning infrastructure services in Clouds

The current Cloud services implement three basic provisioning models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). There are many examples of PaaS and SaaS, which are typically built using existing SOA and Web Services or REST technologies. However, the IaaS model, which intends to provision user or operator manageable infrastructure services, requires a new type of service delivery and operation framework.

## Amazon Web Services

Business model

Provided infrastructure services

Recommended practices and examples

## RackSpace

## Google Application Engine

# Existing standards

This section will include short summary of the standards related to infrastructure services.

## NIST Cloud Computing related standards

The long-term goal of NIST is to provide leadership and guidance around the cloud computing paradigm and catalyze its use within industry and government. NIST aims to shorten the adoption cycle, which will enable near-term cost savings and increased ability to quickly create and deploy safe and secure enterprise solutions. NIST aims to foster cloud computing practices that support interoperability, portability, and security requirements that are appropriate and achievable for important usage scenarios. The NIST area of focus is technology, specifically interoperability, portability, and security requirements, standards, and guidance. The intent is to use the standards strategy to prioritize NIST tactical projects which support USG agencies in the secure and effective adoption of the cloud computing model to support their missions. The expectation is that the set of priorities will be useful more broadly by industry, SDOs, cloud adopters, and policy makers.

### NIST Cloud Computing related activities and Standards

Since first publication of the currently commonly accepted NIST Cloud definition in 2008, NIST is leading wide internationally recognised activity on defining conceptual and standard base in Cloud Computing, which is currently framed out in the following activities:

* NIST Collaboration on Cloud Computing Reference Architecture development
* http://collaborate.nist.gov/twiki-cloud-computing/bin/view/CloudComputing/WebHome
* NIST on Cloud - Standards Acceleration to Jumpstart Adoption of Cloud Computing (SAJACC)
* http://www.nist.gov/itl/cloud/sajacc.cfm
* http://csrc.nist.gov/groups/SNS/cloud-computing/index.html
* NIST Cloud Computing Reference Architecture and Taxonomy
* http://collaborate.nist.gov/twiki-cloud-computing/bin/view/CloudComputing/ReferenceArchitectureTaxonomy

The NIST activity has been resulted in publishing the following documents that create a solid base for cloud services development and offering:

* [NIST CC] NIST SP 800-145, “A NIST definition of cloud computing”, [online] Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
* [NIST CCRA] NIST SP 500-292, Cloud Computing Reference Architecture, v1.0. [Online] http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST\_SP\_500-292\_-\_090611.pdf
* [NIST Synopsis] DRAFT NIST SP 800-146, Cloud Computing Synopsis and Recommendations. [online] Available: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf
* Draft SP 800-144 Guidelines on Security and Privacy in Public Cloud Computing. [online] Available: http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf
* [NIST CC Roadmap] DRAFT NIST SP 800-293, US Government Cloud Computing Technology Roadmap, Volume I, Release 1.0. [online] http://www.nist.gov/itl/cloud/upload/SP\_500\_293\_volumeI-2.pdf
* NIST SP500-291 NIST Cloud Computing Standards Roadmap. [online] Available: http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/StandardsRoadmap/NIST\_SP\_500-291\_Jul5A.pdf

### NIST Cloud Computing Reference Architecture (CCRA)

NIST SP 800-145 document defines Cloud Computing in the following way:

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”

The Cloud Computing as a service type/model includes the following features:

* Five Cloud characteristics
  + On-demand self-service
  + Broad network access
  + Resource pooling
  + Rapid elasticity
  + Measured Service
* 3 service/provisioning models
  + Software as a Service (SaaS)
  + Platform as a Service (PaaS)
  + Infrastructure as a Service (IaaS)
* 4 deployment models
  + Public cloud
  + Private cloud
  + Community cloud
  + Hybrid cloud

For the purpose of this document we also refer to the NIST definition of the IaaS service model:

“The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and *possibly limited control of select networking components (e.g., host firewalls)*.”

Figure 6.1.2.1 presents an overview of the NIST cloud computing reference architecture, which identifies the major actors, their activities and functions in cloud computing. The diagram depicts a generic high-level architecture and is intended to facilitate the understanding of the requirements, uses, characteristics and standards of cloud computing.

As shown in Figure 6.1.2.1, the NIST cloud computing reference architecture defines five major actors: cloud consumer, cloud provider, cloud carrier, cloud auditor and cloud broker. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in cloud computing. Table 6.1.2.1 briefly lists the actors defined in the NIST cloud computing reference architecture and their activities.

|  |  |
| --- | --- |
| Actor | Definition |
| Cloud Consumer | A person or organization that maintains a business relationship with, and uses service from, Cloud Providers. |
| Cloud Provider | A person, organization, or entity responsible for making a service available to interested parties. |
| Cloud Auditor | A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation. |
| Cloud Broker | An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers. |
| Cloud Carrier | An intermediary that provides connectivity and transport of cloud services from Cloud Providers to Cloud Consumers. |

Table 6.1.2.1 Main actors in CCRA and cloud service provisioning

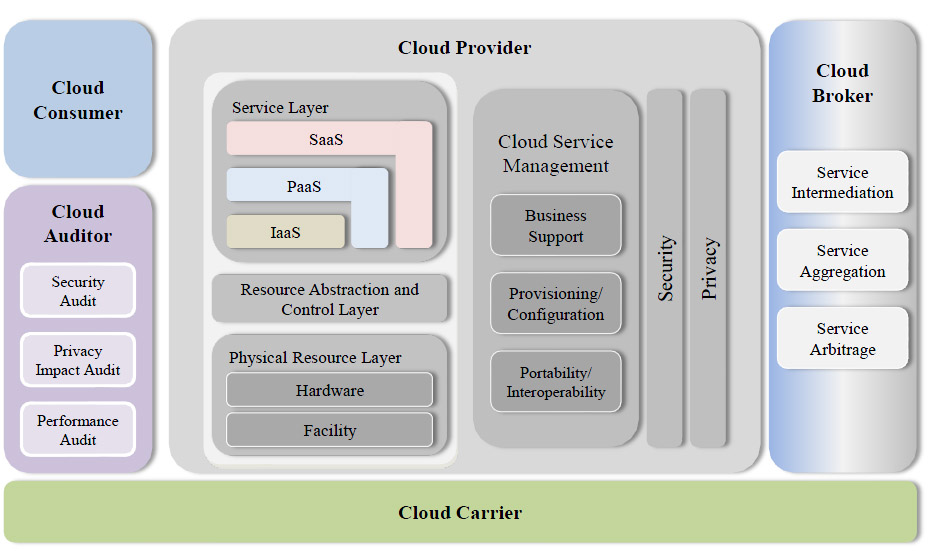


Figure 6.1.2.1 NIST Cloud Computing Reference Architecture (CCRA)

Figure 6.1.2.2 illustrates the interactions among the actors. A cloud consumer may request cloud services from a cloud provider directly or via a cloud broker. A cloud auditor conducts independent audits and may contact the others to collect necessary information. The details will be discussed in the following sections and presented in increasing level of details in successive diagrams.

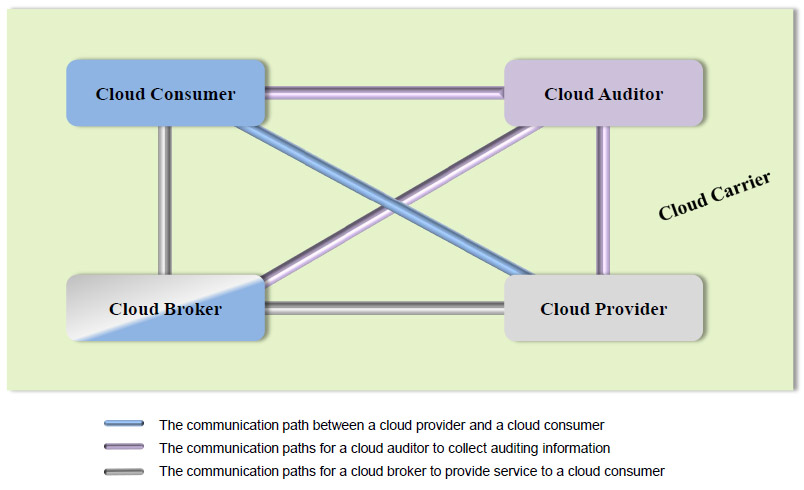


Figure 6.1.2.2 Main role in CCRA service provisioning

#### Relevance and limitation of the NIST CCRA for provisioning network infrastructure as a part of IaaS.

The described above Cloud service model and CCRA are well suited for describing services and business or operational relations, however it is not suitable for design purposes that should allow required functional interfaces between component services and layers.

Despite that current CCRA includes Cloud Carrier as representing typical role of the telecom operators that can provide network connectivity as a 3rd party service, there is no well-defined service model how this can be done. The Cloud Computing definition itself doesn’t require explicitly guaranteed network services. Actually, Cloud computing has been developed primarily for provisioning storage and computing resources and in assumption that network connectivity is provided as ubiquitous Internet connectivity. However, this situation presents serious limitations for large scale use of cloud in enterprise applications that require guaranteed network connectivity QoS and in particular low network latency.

Another problem/limitation of the current CCRA is that it is not suitable for defining required security infrastructure and its integration with main Cloud services or infrastructure that can be potentially multilayer and multi-domain.

The following extension/improvement should be made to at least Cloud IaaS model to meet requirements of the wide range of the critical enterprise services (other service models such as PaaS, SaaS should also allow management of network related parameters):

* Add topology aware infrastructure view
* Provide layered cloud services model that is suitable for defining main inter-layer and inter-service (functional) interfaces
* Add resources and services virtualisation as one of cloud features (TBD)
* Include improved network services definition capable of provisioning required QoS and allowing control from user run applications.

At the business/operational level, the CCRA should be extended to address the following features:

* Better definition of the Cloud Carrier role, operational model and interaction with other key actors
* Extend a set of basic roles with such roles typical for telecom operators/providers as Cloud/infrastructure Operator, customer, and user (as splitting the consumer role)

In connection to above, the provisioned infrastructure services should be characterized and include the following attributes/features:

* Topology definition for infrastructure services including computing and storage resources and interconnecting them network infrastructure
* Corresponding infrastructure/topology description format
* Related topology features and transformation operations (homomorphic, isomorphic, QoS, energy aware etc.)
* (TBD)

In general, the consistent Cloud architecture should address interoperability, compatibility and mobility issues similar how they are addressed in Internet protocol.

## IEEE Intercloud Working Group (IEEE P2302)

IEEE P2302 Working Group recently published a draft Standard on Intercloud Interoperability and Federation (SIIF) [13] that proposes an architecture that defines topology, functions, and governance for cloud-to-cloud interoperability and federation.

Topological elements include clouds, roots, exchanges (which mediate governance between clouds), and gateways (which mediate data exchange between clouds). Functional elements include name spaces, presence, messaging, resource ontologies (including standardized units of measurement), and trust infrastructure. Governance elements include registration, geo-independence, trust anchor, and potentially compliance and audit.

However, the proposed approach has a limited scope by attempting to address a hypothetical scenario when all resources and applications are located and run in multiple clouds and they need to be federated similar to Contend Distribution Network (CDN). The proposed architecture tries to replicate the CDN approach but doesn’t address the generic problems with interoperability and integration of the heterogeneous multi-domain and multi-provider cloud base infrastructure.

The proposed solutions are built around extended use of the XMPP as the proposed base Intercloud protocol and introduce Intercloud Root and Exchange Hosts to support Intercloud communications, trust management and identity federation.

The limitation of the proposed architecture and approach is that it tries to closely imitate Internet approach in building hierarchical interconnected infrastructure for Internet protocol based services to support inter-cloud communication. But actually there is no need for such additional inter-cloud layer to provision and operate cloud based infrastructures because when provisioned the cloud based infrastructure can run as normal infrastructure using all Internet technologies directly to support intra- or inter-provider communications, given the appropriate network virtualisation and address translation technologies are used. Cloud technologies provide a virtualisation platform for IT and network services and allow for the entire infrastructure instantiation together with related protocols and core infrastructure services related to control and management functions.

## IETF

Clouds with their service oriented and utility/infrastructure focused approach are not generally fall into the IETF focus, however a number of initiatives at IETF made attempts to address network and protocol related issues in cloud operation and inter-cloud interoperability. Currently, cloud infrastructure related issues are considered in a number of WGs related to address allocation (), CDN inter-connection (BoF ), incidents reporting and attacks protection (BoF ).

Cloud Bar BoF initially took place at IETF78 (August 2010) and have been resulted in submission of a few Internet Drafts providing information for Internet community on the current Cloud Computing standardisation and practices and proposing a general Cloud and Inter-Cloud frameworks, in particular:

* Cloud/DataCenter SDO Activities Survey and Analysis, version 0.2, December 28, 2011 (draft khasnabish-cloud-sdo-survey-02.txt)
* Cloud Reference Framework. Internet-Draft, version 0.2, December 27, 2011 ([draft-khasnabish-cloud-reference-framework-02.txt](http://www.ietf.org/id/draft-khasnabish-cloud-reference-framework-02.txt))

The document intends to propose a framework for developing possible new standards to consistently support Cloud services with the network, transport and messaging services which are in the scope of IETF standardisation. The document defines the Cloud Reference Framework and Inter-Cloud Framework. The proposed Cloud Reference Framework defines a number of horizontal layers:

* Data/Content Layer(DCL)
* Application/Service Layer(ASL)
* Resource Control Layer(RCL)
* Resource Abstract and Virtualization Layer(RAVL)
* Physical Resource Layer(PRL)

and one vertical plane:

* Cloud Management Plane.

The draft also intends to provide the definition of the Inter-Cloud Framework that should address the following requirements:

* Provide a mechanism for resource search and discovery,
* Provide a mechanism to authenticate participating entities,
* Provide a mechanism for requesting, controlling, and releasing resources between two clouds.
* Provide a secure transport channel between the interconnecting entities.
* Provide end-to-end isolation to support multi-tenancy.
* Provide a mechanisms for monitoring, QoS, assuring, and troubleshooting across the interconnection.

Possible inter-Clouds Interfaces may include provisioning, signaling, control, monitoring, management, transport, security, naming and addressing.

Although this draft provides a good definition of the network related functionalities in typical cloud infrastructures, it doesn’t consider the cloud infrastructure services as an integrally management component.

c) Cloud Service Broker, Internet-Draft, version 0.3, March 26, 2012. (draft-shao-opsawg-cloud-service-broker-03.txt)

This document introduces a Cloud Service Broker (CSB) entity to provide brokering functions between different Cloud Service Providers and Cloud Service consumers.

In the cloud ecosystem, the Cloud Service Requesters/Consumers can through CSB to access cloud computing services and resources of Cloud Service Providers. When CSB receives the cloud service consumer requests from Cloud Service Requesters/Consumers, it will select appropriate cloud computing services and resources by Cloud Service Providers and specific function pattern to execute related cloud service operations such as intermediation, proxy, monitoring, transformation/portability, governance, provisioning, screening, substitution, security, composition services. CSB will invoke and adapt to the concrete cloud services and resources from various Cloud Service Providers, and return consumer response to Cloud Service Requesters/Consumers.

According to SOP Architecture [SOP Architecture]document, CSB is similar to the proxy and specific Cloud Service Providers' platform is similar to Service Node in a SOP Network Architecture.

## ITU-T Focus Group Cloud Computing

The ITU-T Focus Group Cloud Computing published a number of documents that analyse and discuss cloud technologies and components from the point of view of telecom industry and telecom providers.

<http://www.itu.int/en/ITU-T/focusgroups/cloud/Pages/default.aspx>

== To be extended ===

## Related activities at OGF

OGF has a number of activities that are related to Infrastructure Services required by and provisioned for Grid and Cloud based applications. This section provides short overview of such activities and suggestions how they can be used for making ISOD systems interoperable.

### OCCI – Open Cloud Computing Interface

OCCI began in March 2009 and was initially lead by co-chairs from the once SUN Microsystems, RabbitMQ and the Universidad Computense de Madrid. Today, the working group has a membership of over 250 members and includes numerous individuals, industry and academic parties. Some of these members that have contributed include:

* Industry: Rackspace, Oracle, Platform Computing, GoGrid, Cisco, Flexiscale, ElasticHosts, CloudCentral, RabbitMQ, CohesiveFT, CloudCentral.
* Academia & Research projects: SLA@SOI, RESERVOIR, Claudia Project, OpenStack, OpenNebula, DGSI.

The reasons driving the development of OCCI are:

* Interoperability – Allow for different Cloud providers to work together without data schema/format translation, façade/proxying between APIs and understanding and/or dependency on multiple APIs
* Portability – No technical/vendor lock-in and enable services to move between providers allows clients easily switch between providers based on business objectives (e.g. cost) with minimal technical cost and enables and fosters competition.
* Integration – Implementations of the specification can be easily be integrated with existing middleware, 3rd-party software and other applications.
* Innovation – Driving modern technologies.

### Network Service Interface Working Group (NSI-WG)

NSI started in OGF24 (2008)

Short description of the main NSI-WG concepts and documents and how they can be used to support ISOD, in particular:

* Common NSI that should be supported by ISOD and NRPS systems
* Several demonstrations in 2011 (i.e. GLIF meeting in September, FIA in October, and SC’11 in November) has shown the magnitude of the NSI CS protocol, allowing inter-operability with various provisioning systems (i.e. AutoBAHN, G-Labmda, OpenNSA/Argia, and DRAC) on a global testbed.

### Network Markup Language Working Group (NML-WG)

# Existing Cloud Middleware for Infrastructure Services Provisioning

The existing Cloud Management software (middleware) provides a functional layer between a user or user application and physical resources which are typically virtualised to allow cloud based services virtualisation and provisioning on-demand with dynamic scaling or elasticity.

## OpenNebula

## OpenStack

## Eucalyptus

# Projects on Infrastructure Services Provisioning – To be discussed

# Proposed Taxonomy and Analysis

# Summary and Recommendations

This section will provide recommendation about using described best practices and solutions.

# References

1. Amazon Web Services. [Online] Available at <https://aws.amazon.com/>
2. GoGrid. [Online] Available at <http://www.gogrid.com/>
3. OpenStack. [Online] Available at <http://www.openstack.com/>
4. Rackspace. [Online] Available at <http://www.rackspace.com/>
5. VMware. [Online] Available at <http://www.vmware.com/>
6. Amazon Beanstalk. [Online] Available at <http://aws.amazon.com/elasticbeanstalk/>
7. CloudFoundry. [Online] Available at <http://www.cloudfoundry.org/>
8. Google AppEngine. [Online] Available at <http://code.google.com/appengine/>
9. Microsoft Windows Azure. [Online] Available at <http://www.microsoft.com/windowsazure/>
10. SaleForce Force.com. [Online] Available at <http://www.heroku.com/>
11. Cloud9 Analytics. [Online] Available at <http://www.cloud9analytics.com/>
12. CVM Solutions. [Online] Available at <http://www.cvmsolutions.com/>
13. GageIn. [Online] Available at <http://www.gagein.com/GuestHome>]
14. KnowledgeTree. [Online] Available at <http://www.knowledgetree.com/>
15. [Online] Available at <http://www.sjaaklaan.nl/pivot/entry.php?id=142>
16. AutoBAHN. [Online] Available at <http://autobahn.geant.net>
17. Willner, A., C.Barz, J.Garcia-Espin, J.F.Riera, S.Figuerola. *Harmony: Advance reservation in heterogeneous multi-domain environment*. Proceedings of the 8th IFIP Networking conference, Springer's LNCS, 5 2009. ISBN: 978-3-642-01398-0
18. Guok, C., D..Robertson, E.Chaniotakis, M.Thompson, W.Johnston, Brian Tierney, *A User Driven Dynamic Circuit Network Implementation*, Proceedings DANMS2008 Conference, IEEE, July 2008.
19. GEANT Project. [Online] Available at http://www.geant.net/pages/home.aspx
20. Generalised Architecture for Dynamic Infrastructure Services (GEYSERS Project) [Online] Available at http://www.geysers.eu/
21. Phosphorus Project. [Online]. Available at http://www.ist-phosphorus.eu/
22. Amazon Web Services: Overview of Security Processes. November 2009. http://aws.amazon.com/security
23. European Grid Infrastructure (EGI). [Online] Available at https://www.egi.eu/
24. Cloud/DataCenter SDO Activities Survey and Analysis, version 0.2, December 28, 2011. <http://tools.ietf.org/id/draft-khasnabish-cloud-sdo-survey-02.txt>
25. Cloud Reference Framework. Internet-Draft, version 0.2, December 27, 2011. <http://www.ietf.org/id/draft-khasnabish-cloud-reference-framework-02.txt>
26. Cloud Service Broker, Internet-Draft, version 0.3, March 26, 2012. <http://tools.ietf.org/id/draft-shao-opsawg-cloud-service-broker-03.txt>