GFD-I.???
ISOD-RG

Version: 1.3
Late Updated: 8 Feb 2013

Editor(s)
Chin Guok (ESnet)

Contributors
Pawel Brzozwski (ADVA)
Scott Campbell (CRC)
Tangui Coulouarn (Forksningsnettet)
Yuri Demchenko (UvA)
Freek Dijkstra (SARA)
Michal Giertych (PSNC)
Joan Antoni Garcia Espin (i2CAT)
Eduard Grasa (i2Cat)
Chin Guok (ESnet)
Jeroen van der Ham (UvA)
Radek Krzywania (PSNC)
Tomohiro Kudoh (AIST)
Mathieu Lemay (Inocybe Technologies)
Atsuko Takefusa (AIST)
Alexander Willner (Univ. of Bonn)
Yufeng Xin (RENCI)

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

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 of 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).

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 adds to the underlying virtualized infrastructure 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 package 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] .

A critical component of cloud provisioning that is often overlooked is the network connectivity aspect of the system. With the ubiquitous nature of cloud computing, networks are starting to play a more vital role than simply a supporting infrastructure utility such as power and water. Network predictability and guarantees are emerging requirements necessary to effectively implement a cloud service. To meet these demands, it is necessary to view networks not simply as a (1) data plane, where data is transported; but also a (2) control plane, where path computation and signalling functions associated with the control of the data plane is performed; and a (3) management plane, where systems and processes to monitor, manage, and troubleshoot the network reside.

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.

2. Infrastructure Services definition

2.1 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, in reference to their Integrated Information Infrastructure Reference Model (III-RM) [15] , 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

With 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 [16] , 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/)*"*

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"*

2.2 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 be an 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.)

3. 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.

3.1 On Service Provisioning

Imagine you own a business, of any type, offering your customers services. Now, customers are very important as they are the ones funding your business. In order to thrive, you need to win them and maintain them. Hence you try to address their needs to the fullest. So, what do your customers care about? They want the service to be a) good, b) cheap and c) fast. It's just that simple.

As grotesque and obvious the paragraph above may sound, it is crucial to remember those principles when trying to address any networking problem with a technical solution. Telco business is no different from other service offering businesses. ISPs want to have their network services to be:

a) *good* – reliable and meeting any expectances customer might have with regard to traffic,
b) *cheap* – with the lowest OPEX and CAPEX possible,
c) *fast* – easily and quickly accessible by the customers.

IP networks meet those requirements very well. They are easy to configure and maintain, allowing for low OPEX. Given proper design, they can offer services meeting almost any QoS requirements. Their simplicity allows NOC engineers to track errors easily and hence react to network failures in a fast pace, increasing reliability. IP flexibility allows for easy adaptation to customer needs. But foremost, thanks to statistical multiplexing, they allow for bandwidth overprovisioning, allowing for utilizing invested CAPEX to the fullest. IP technology makes a great service provisioning platform.

On the other hand, transport networks, when evaluated from servicing user's perspective look very dim. They are all but easy to configure and require highly skilled engineers to operate and troubleshoot the network. Lacking statistical multiplexing, they cannot take advantage of dynamic traffic patterns in the network. Lack of flexibility from IP layer puts them as ill-fitted as a user servicing layer in most cases. They offer several advantages over IP though, being longer transmission reach and number of bits per $ transported over a link.

Aforementioned reasons are why so many networks are designed as a set of interconnected IP routers and switches. Operators often treat transport technologies as a necessary must and reach out to them primary if routers are too far away from each other or to increase the capacity of the link between routers via adding some form of WDM multiplexing technique. Transport domain boundaries are hence limited to scope of a link. Such networks are easy to maintain and offer great OPEX and fair CAPEX. Service provisioning in such networks is straightforward and NRPSes used are often limited to router command line interfaces.

The challenge with such network architecture is how it scales with amount of supported traffic in terms of CAPEX. With the constant increase of amount of data transported by each service, traffic flowing through a network drastically increases. This can be addressed by adding new line cards to the routers and/or upgrading to ones with higher throughput. As the traffic requirements advance, so does the technology, offering line interfaces with much more throughput for some CAPEX increase. As a matter of fact, we are experiencing such advancement at the very moment, with carriers deploying 100G in place of 10G line interfaces. While it's easy to economically justify upgrading throughput of line interfaces at the edges of the network, it's tougher to do so for line interfaces used on the transit nodes. Customers will pay more if sending more traffic, hence justifying network edge investments. But the customer does not care much on how the traffic is sent within the ISP's network and he will not be willing to pay more to fund the ISP's

expenditures on transit nodes. Hence there is a trade-off to be made with this architecture – while offering arguably the best OPEX, with the amount of traffic increasing, CAPEX can become considerable.

The need of continuous expenditures on the transit nodes can be also addressed in a different manner – by expanding the transport domain scope between the routers, often called optical bypass (as the transport technology used is most often xWDM one). In such an approach, routers can be interconnected with each other by the underlying transport network in many possible ways, depending on the IP connectivity needs. Hence transit switching on IP nodes is delegated to the optical layer. As optical switching is orders of magnitude less expensive than the IP switching, this solution offers great CAPEX savings. There are a number of challenges related to this solution – service provisioning and troubleshooting becomes more complicated in such networks, increasing OPEX. As mentioned before, the optical layer is also much less flexible than the IP layer, hence given improper design and with traffic patterns varying in the network, IP switching is still necessary to mitigate these issues. In extremes, this can lead to substantially worse utilization of resources (line cards), than in the scenarios where the transport network was link scoped (i.e consistent physical (optical) and logical (IP) topology). These challenges can be addressed by NRPSes, aiming to increase the ease of configurability and flexibility of the optical layer. An example of a NRPS, allowing for IP+optical integration via transport network virtualization is discussed below.

3.1.1 Virtual Optical Networks

Consider a network in Figure 3.1.1, where IP routers are connected via colored interfaces directly to optical nodes (OXCs). The optical layer is built with components allowing flexibility in switching, e.g. colorless and directionless ROADMs. As such, IP routers could be potentially interconnected in any manner with each other by proper optical resource provisioning. This potential connectivity is pre-planned by the operator and presented to the IP layer as *virtual links*.

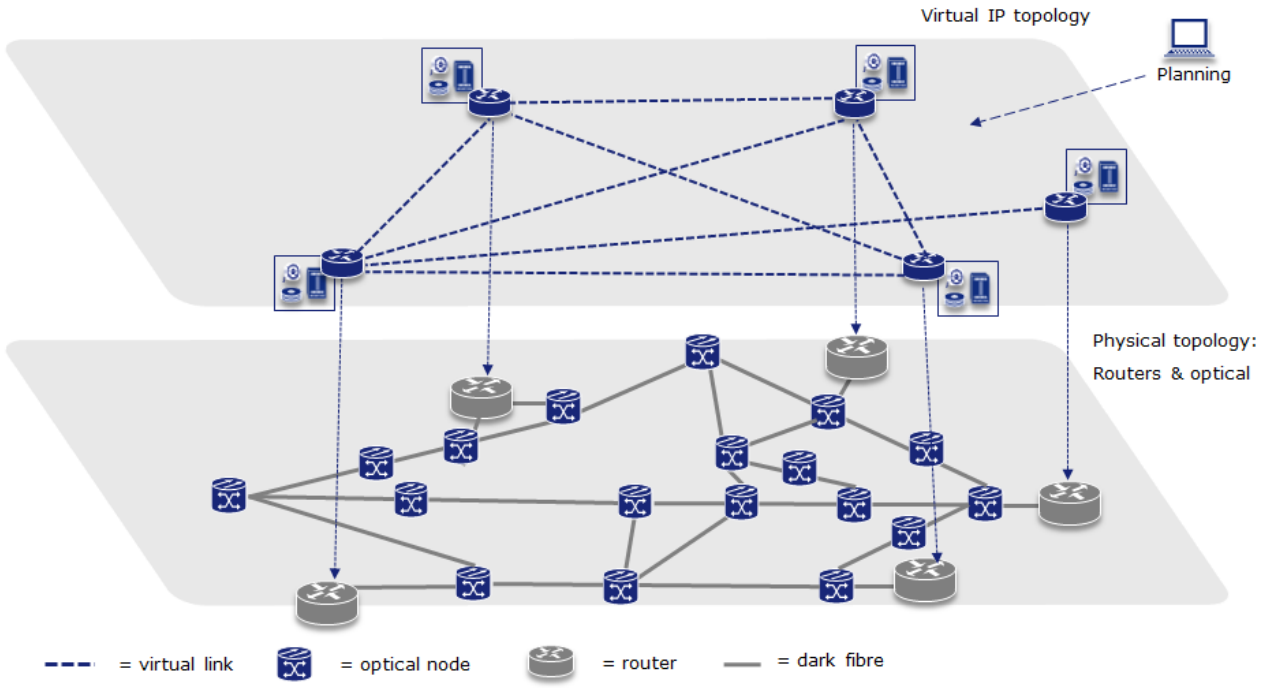

Figure 3.1.1 Optical Network Virtualized

It is important to note that it is not necessary to have actually provisioned the specific resources in the optical network corresponding to a given virtual link in IP network in order for the link to be made available to the IP network. A virtual link is simply an indication of the potential connectivity within the server layer network. Only when the resources on the link are actually reserved during subsequent signalling/provisioning operations for the end-to-end IP service is it necessary to also provision the underlying server layer network resources - using an NRPSes allowing for hierarchical service activation capabilities, e.g. implementing a [GMPLS E-NNI] interface.

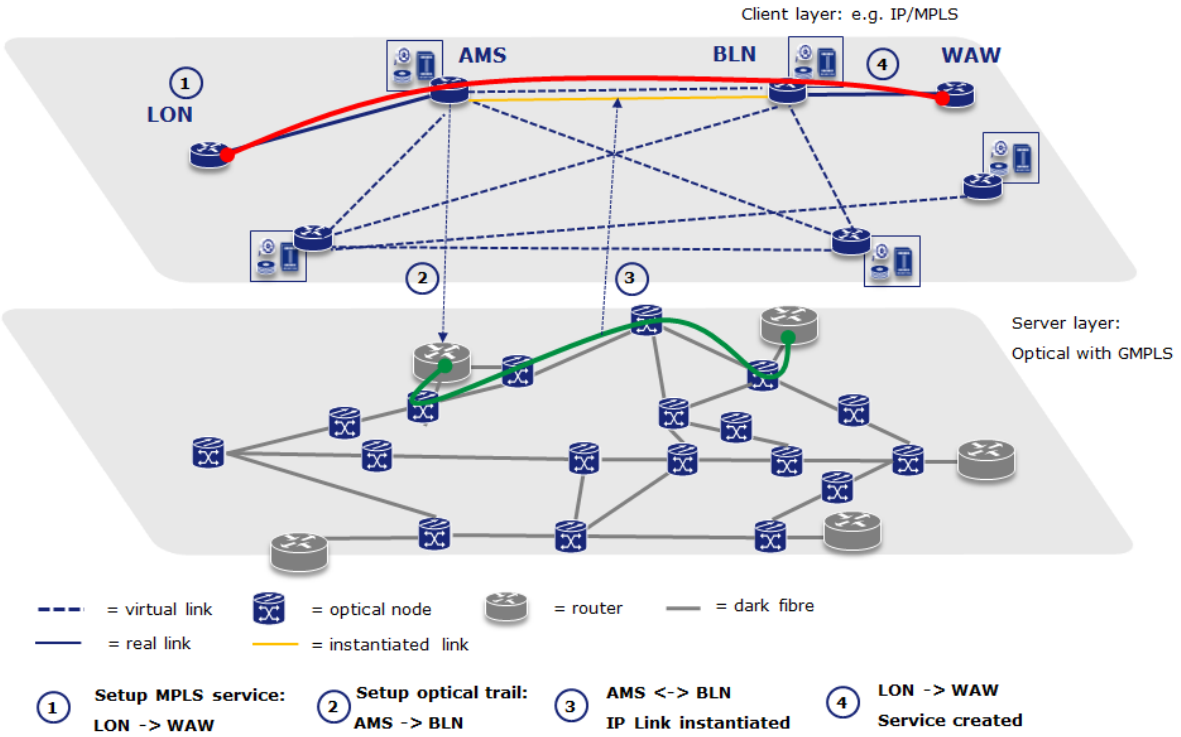

Figure 3.1.2 Hierarchical service activation with virtualized optical networks

If an end-to-end MPLS service was to be created from London to Warsaw, provisioning sequence will be as follows:

1. LON router will compute a path using traffic engineering information from IP layer. He will include both real links (e.g. LON->AMS) and virtual ones in this computation. As a matter of fact, LON will most likely not be able to differentiate between virtual and real links, hence treating them the same. A {LON, AMS, BLN, WAW} path will be computed to facilitate this service. IP NRPS will be used to setup this service. Service setup could be signaled to AMS e.g. via RSVP.
2. Once resource provisioning in IP reaches the AMS router, AMS will suspend IP service setup and request AMS->BLN optical trail setup via hierarchical service activation mechanism. The optical trail will be set up by an underlying optical NRPS (e.g. GMPLS)
3. Once optical trail is set up, IP layer will be notified about this fact - virtual link will be hence *instantiated*, i.e. will become a real link.
4. IP service setup will be unsuspended on AMS and signaled up to WAW.

3.2 Argia

Argia is an IaaS framework based product to create IaaS 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 3.2.1 shows the network elements supported by the current release of Argia (Argia 1.4). Table 3.2.2 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 3.2.1 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 3.2.2 Current and past Argia deployment

Argia's software architecture, depicted in Figure 3.2.1, 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).

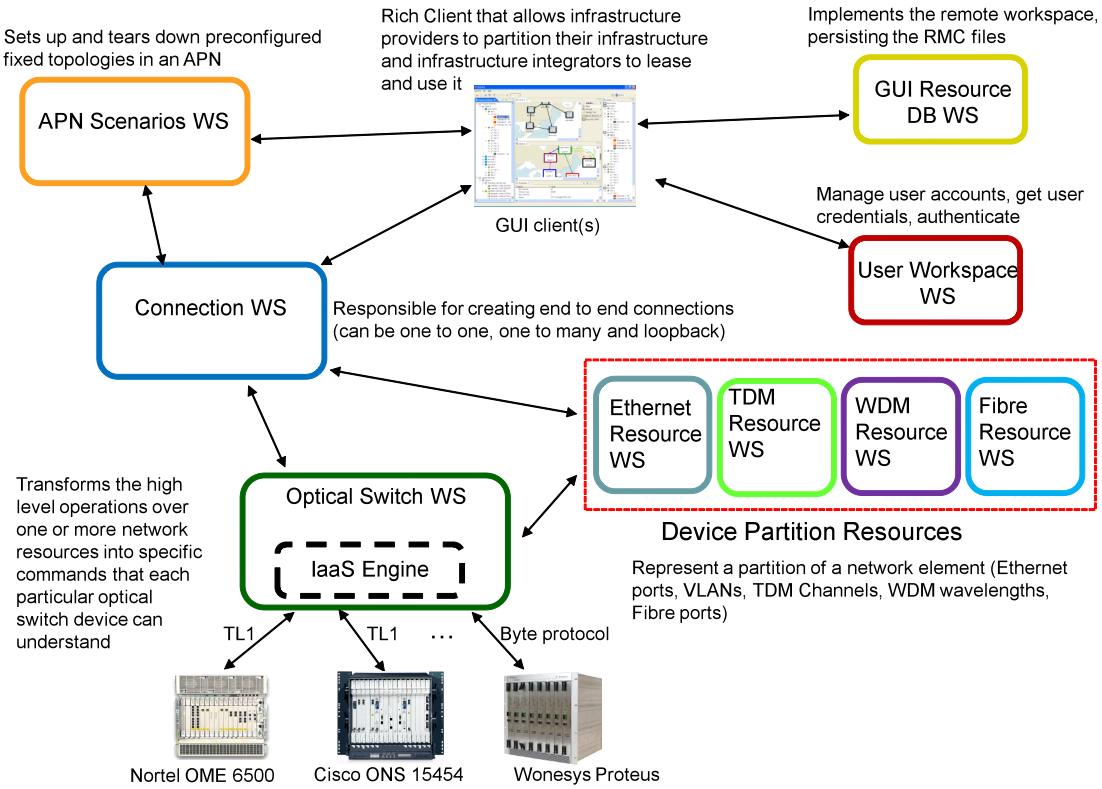

Figure 3.2.1 Argia 1.4 Service Oriented Architecture

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.

3.3 AutoBAHN

The AutoBAHN [17] tool was conceived during the GÉANT2 project (2005-2009), and continues in the GÉANT3 project (2009-2013) [18] . The objective was to create a generic multi-domain system that was be able to integrate Europe's heterogeneous National Research and Education Network (NREN) infrastructures and could be used in the GÉANT Bandwidth on Demand (BoD) Service [19] . 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 3.3.1.

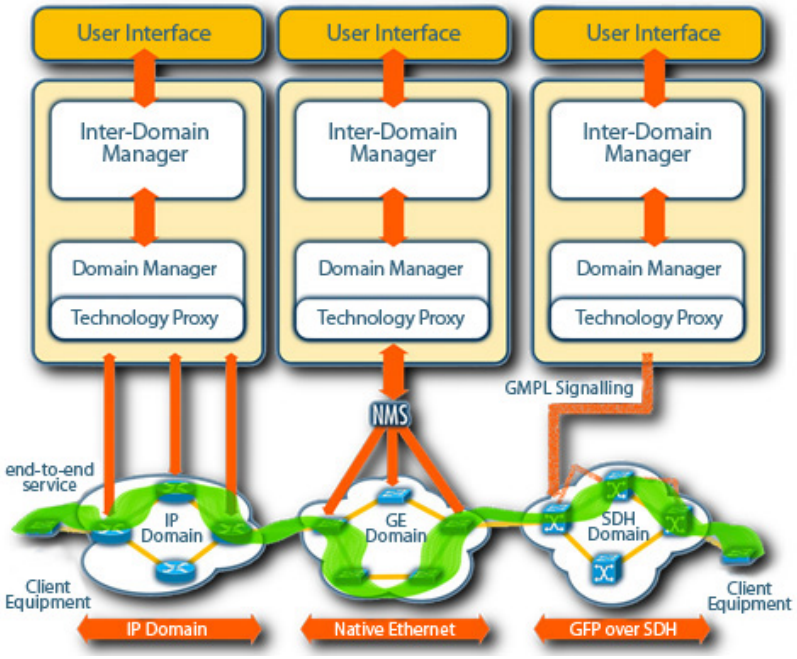

Figure 3.3.1 AutoBAHN architecture overview
(figure from http://www.geant.net/service/autobahn/User_Experience/Pages/UserExperience.aspx)

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 3.3.2 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.

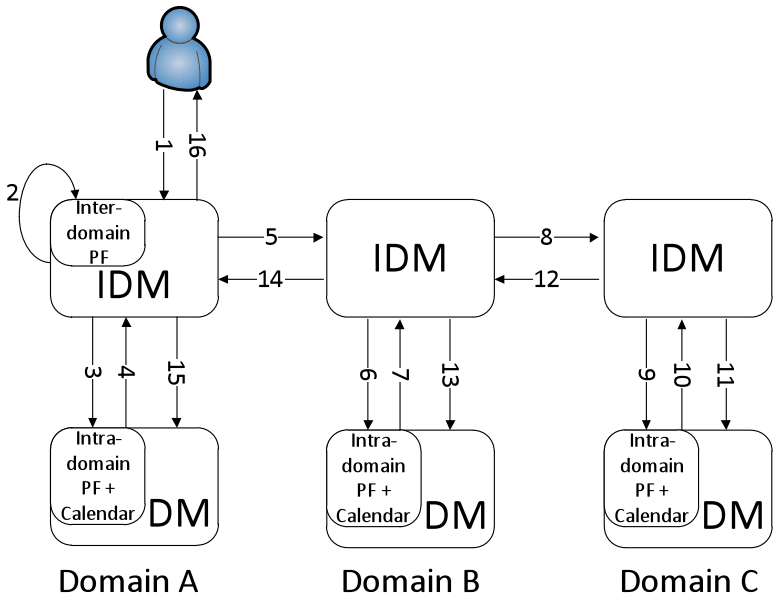

Figure 3.3.2 AutoBAHN circuit request processing

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 overhead. 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 3.3.3). 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.

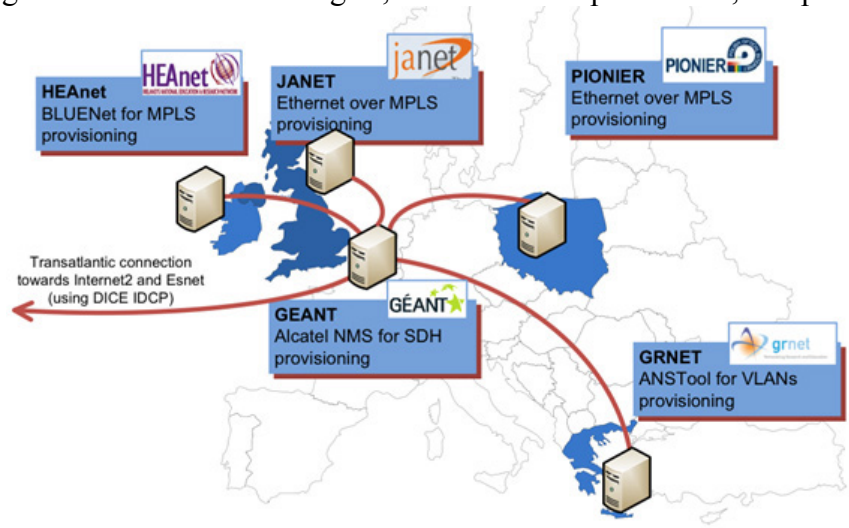

Figure 3.3.3 AutoBAHN pilot deployment
(From: http://www.geant.net/service/autobahn/User_Experience/Pages/UserExperience.aspx)

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.

3.4 G-lambda and GridARS

The G-lambda [20] project, started in 2004 as a collaboration between Japan's industrial and governmental laboratories, KDDI R&D Laboratories, NTT, NICT and AIST, had the goal of defining a Web service-based network service interface, named GNS-WSI (Grid Network Service - Web Service Interface), through which users or applications could request end-to-end bandwidth-guaranteed connections.

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.

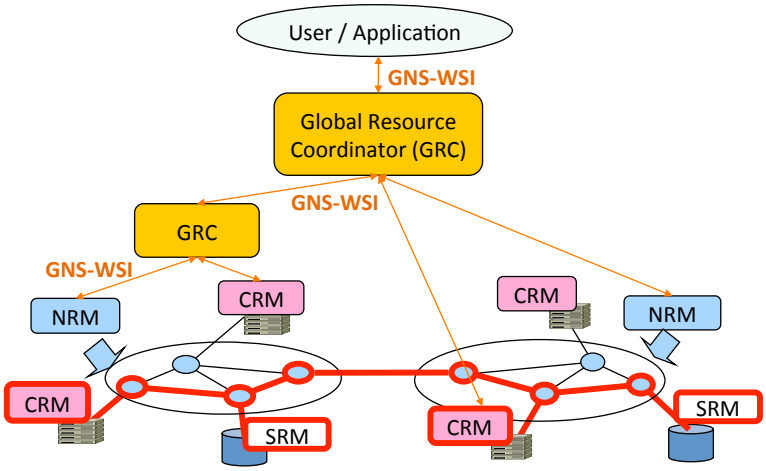

Figure 3.4.1 Reference model of GNS-WSI3

Figure 3.4.1 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 3.4.1 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 3.4.1. 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 3.4.1 GNS-WSI3 operations

AIST has been developing a GNS-WSI reference implementation, called the GridARS [21] 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.

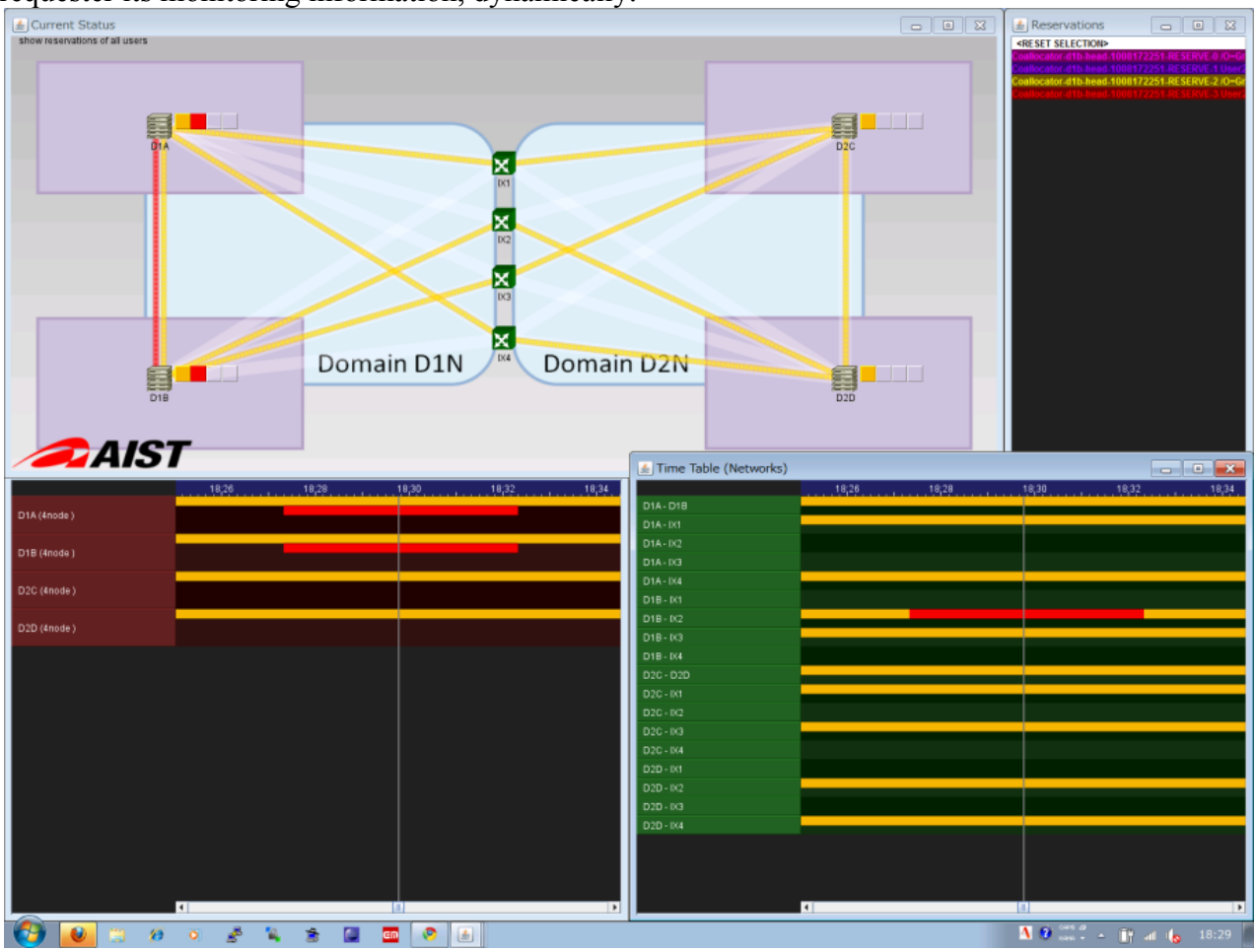

Figure 3.4.2 GridARS reserved resource status

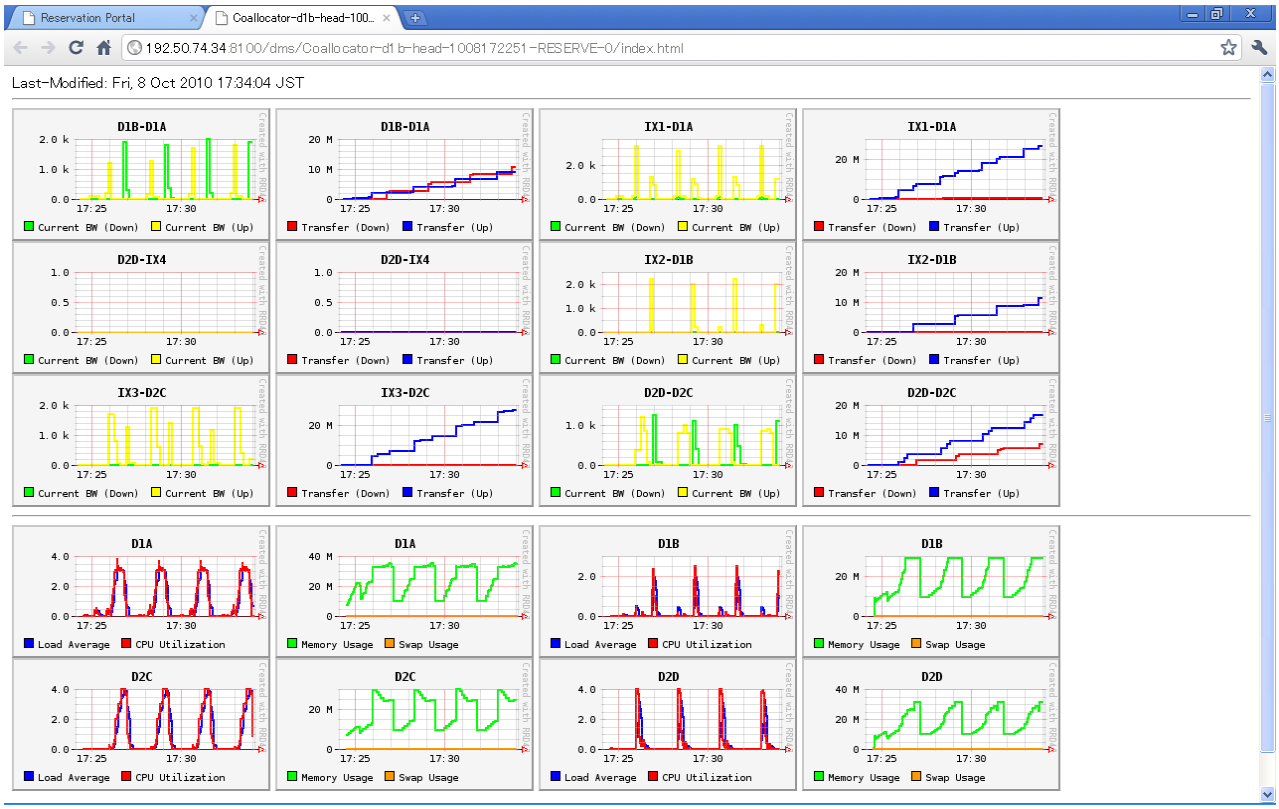

Figure 3.4.3 GridARS network and computer resource status of the virtual infrastructure

KDDI R&D Laboratories and NTT have also developed their own reference implementations of GNS-WSI. This was demonstrated at iGrid2005 [22] as a single domain by using GNS-WSI v.1 and international multiple domains at GLIF2006 [23] and GLIF2007 by using v.2. G-lambda was also involved in the Fenius [24] and OGF NSI [25] interoperation demo in 2010 and 2011 using GNS-WSI v.3.

3.5 OSCARS

The On-demand Secure Circuits and Advance Reservation System (OSCARS) [26] 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 [27] :

- *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.

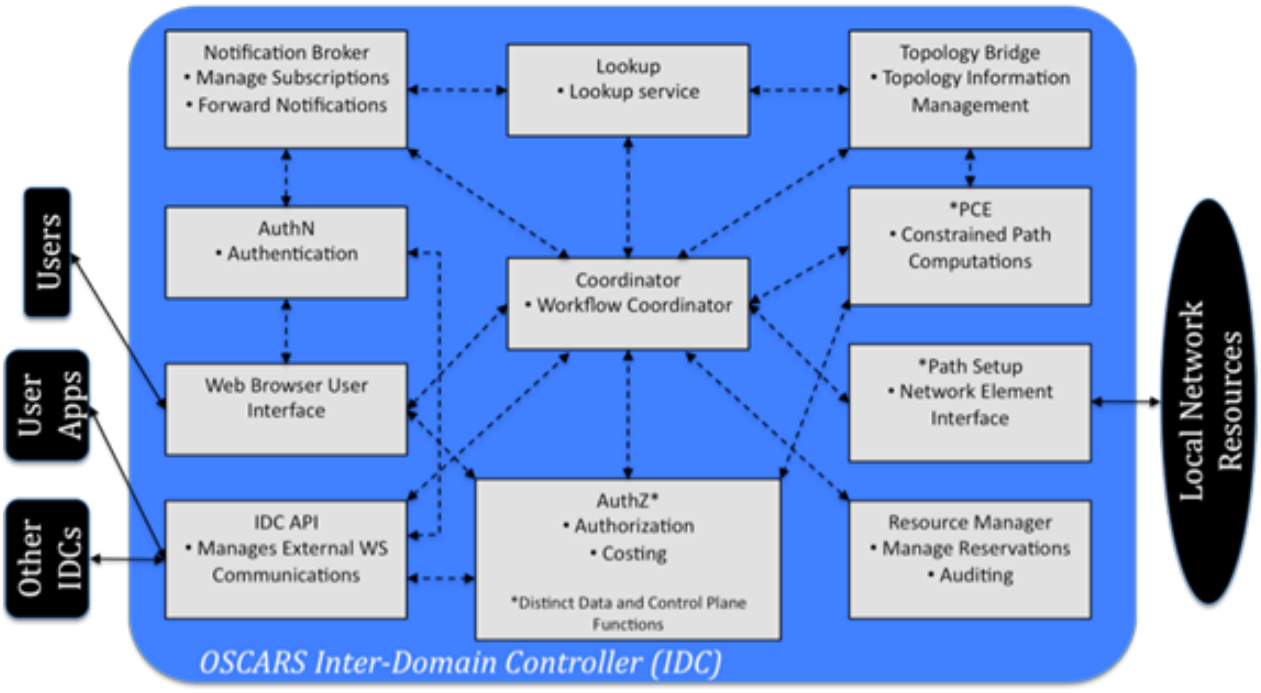

Figure 3.5.1 OSCARS software architecture

- *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) to the subscriber. In addition to

circuit service users, the notification broker is used to notify the perfSONAR [28] 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 [29] 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.

4. General and Cloud Oriented Network Infrastructure Services Provisioning

4.1 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 [30] [31] . Based on the extended 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 intercloud 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.

4.1.1 ORCA Architecture and 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 [31] ; (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.

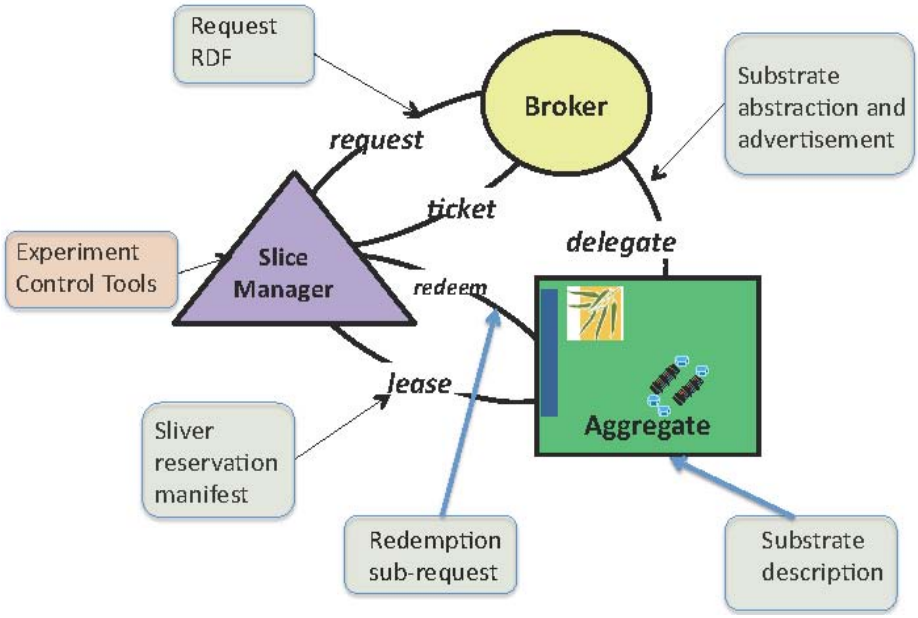

Figure 4.1.1.1 Information flow life cycle in ORCA architecture

4.1.1.1 *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.

4.1.1.2 *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.

4.1.1.3 *Slice reservation model*

This is the abstract model used by ORCA brokers to return resource tickets to the SM controller. Each ticket contains 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.

4.1.1.4 *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.

4.1.2 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) [32] 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 [33] , to create and maintain the ontologies. The Jena RDF/OWL Java package for Java [34] , 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.

4.1.3 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 [30] [35] a Eucalyptus (and now OpenStack) extension that allows guest VIRTUAL MACHINE 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 virtual machines 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.

4.1.4 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 end-to-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.

4.2 GEYSERS Generalised Infrastructure Services Provisioning

4.2.1 GEYSERS Architecture

GEYSERS [36] 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 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.

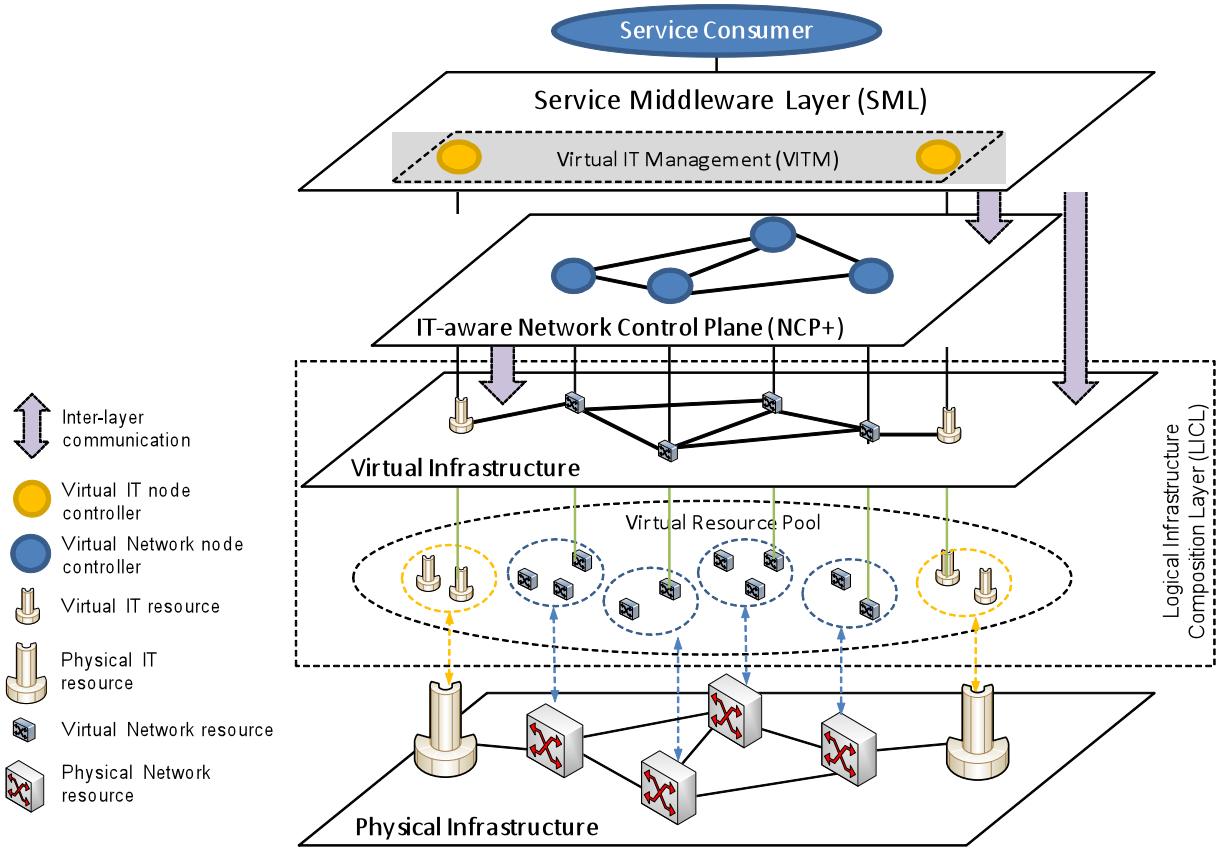

Figure 4.2.1.1 GEYSERS layered architecture

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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

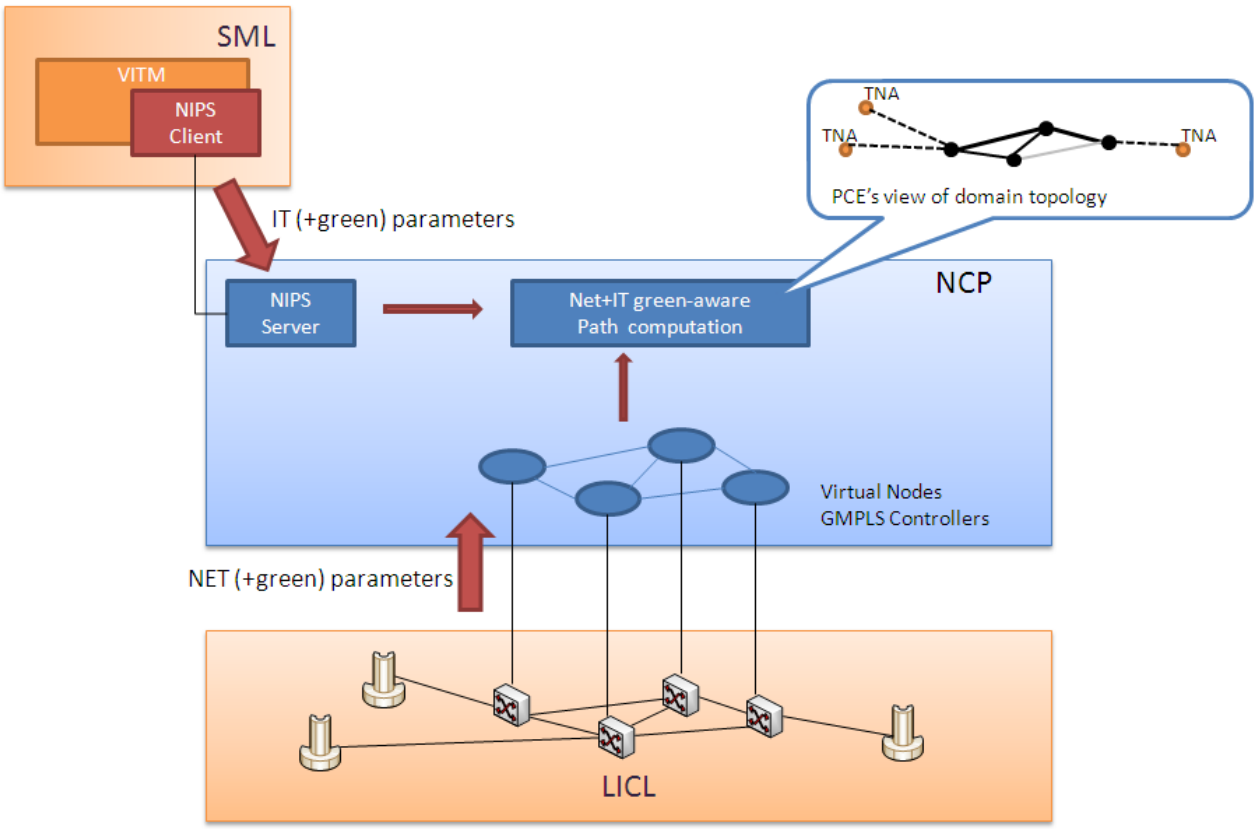

Figure 4.2.4.1 NCP – Network and IT energy-aware service provisioning

The NCP+ is based on the ASON/GMPLS [37] and PCE [38] 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 4.2.4.1 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).

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.

4.3 OpenNaaS: An Open Framework for Networking as a Service

While dynamic and multi-domain provisioning of network resources has been a long-standing research area, the rise of cloud computing now imposes even more challenging requirements. Very often, organizations do not have the capability to manage their network infrastructures, whether to build new ones, or to have tools to manage their part of the network as it has remained under the control of the infrastructure providers. These organizations need a solution that offers them an Infrastructure as a Service with the capability to configure or deploy their own services on top.

In order for NRENs and operators to be able to deploy and operate innovative NaaS offerings, an appropriate toolset needs to be created. With such goal in mind, Mantychore FP7 [39] has created the OpenNaaS framework [40] .

OpenNaaS was born with the aim to create an open source software project community that allows several stakeholders to contribute and benefit from a common NaaS software stack. OpenNaaS offers a versatile toolset for the deployment of NaaS oriented services. The software is released with a dual L-GPL/ASF licensing schema that ensures that the platform will remain open and consistent, while commercial derivatives can be built on top. This open schema allows trust to be built on the platform, as NRENs and commercial network operators can rely on the continuity, transparency and adaptability of the platform.

In that sense, each network domain would be able to use their own OpenNaaS instance to:

- Get resources from the network infrastructure: routers, switches, links or provisioning systems.
- Abstract them to service resources, independently of vendor and model details.
- Embed instantiation of virtualized resources in the regular BSS workflows.
- Delegate management permissions over the infrastructure resources they own so that "Infrastructure integrators" can control them during a period of time.

With an eye on versatility and smooth integration, OpenNaaS offers a powerful remote command line, as well as web-service interfaces. This web-service interface will offer the possibility to both build a GUI and integrate it with existing middleware applications already deployed in the virtual research organisations.

4.3.1 OpenNaaS Extensions

Several extensions for OpenNaaS are being developed inside the Mantychore FP7 project. Beyond that, other extensions are being created in parallel projects, leveraging the same core components and bringing reusability across distinct research efforts. As of version 0.9, OpenNaaS supports the following abstracted resources:

- Routers
- ROADM's
- BoD Domain services

- IP Networks

For the Router resource, a driver for JunOS has been implemented which supports virtualization (interfaces and routing instances), IP, routing (static and OSPF) and GRE tunnels. More capabilities will be added in the near future, such as IPv6 and firewalling. The ROADM extension allows dynamically modifying the ROADM switching table between optical channels.
On the other hand, the BoD Domain represents an opaque domain, controlled by a provisioning system that the user can call to obtain L2 links. The current implementation supports GEANT BoD (AutoBAHN).

The IP Network resource groups together Routers and BoD Domain resources into a logical unit, while adding a topology. This allows users to manage all the resources in an orchestrated way (currently you can deploy network-wide OSPF configurations). Furthermore, it allows delegating configuration rights to users in an aggregated mode, instead of dealing with individual infrastructure resources.

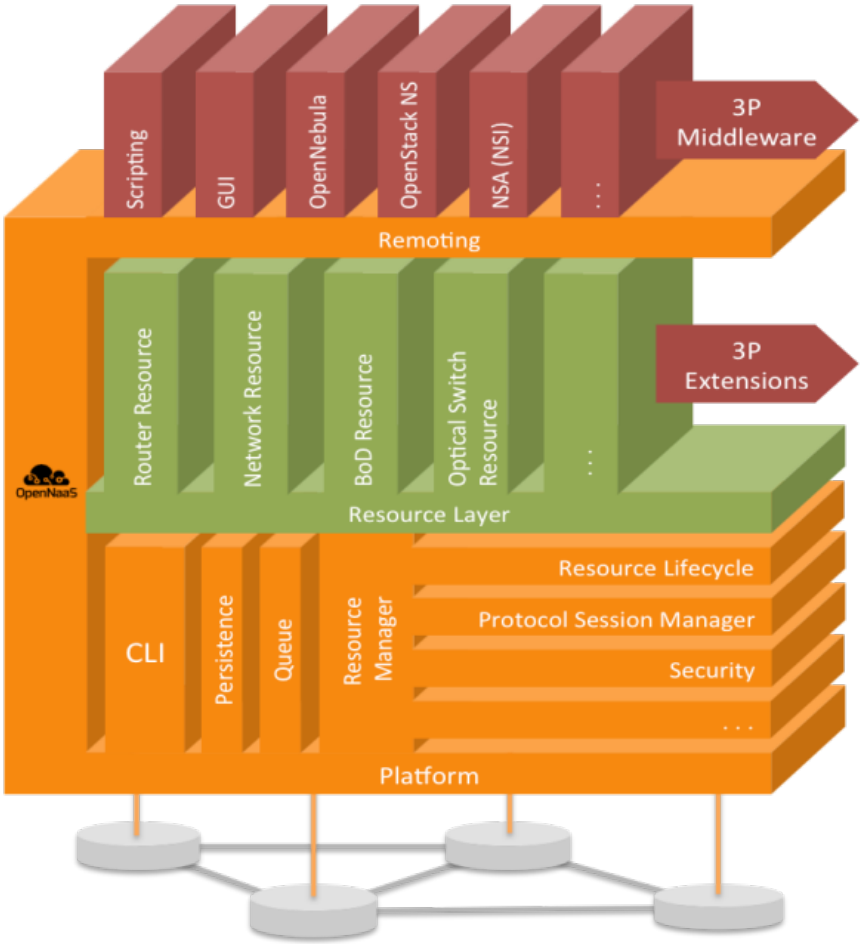

Figure 4.3.1.1 OpenNaaS Framework Architecture. Three main parts can be distinguished: the Platform Layer (bottom), which contains all common functions for resource management and device driving; the Resource Layer (middle), where the different resource models with associated capabilities are implemented; and the Remoting Layer (top), where all connectors to 3rd party middleware systems are implemented.

4.3.2 The OpenNaaS Community

The OpenNaaS user community can be split in three different roles:

- **Infrastructure Providers**: Those who own a network infrastructure (NRENS or telecom operators) and need enhanced flexibility in order to lease virtualized slides.
- **Service Providers**: While traditionally overlapping with the infrastructure provider, this emerging role is composed of those who aggregate third party virtual infrastructure resources, in order to add value for concrete customers or vertical markets. A service provider needs to maintain tight control on what resources are being consumed and the management rights delegated to end-users.
- **End-User**: The NaaS end-user has sophisticated needs which go beyond the pure consumption of the networking infrastructure. Complex and demanding applications not only need a certain degree of management control over the resources, but also need quick and on demand provisioning.

With this in mind, the following user groups can benefit from OpenNaaS:

- Network and datacenter operators who seek to provide richer virtualization services to their customers.
- Infrastructure management software, i.e. cloud managers, provisioning systems or simple software projects which can benefit from extended infrastructure management and awareness.
- Research groups or individuals who would like to use a virtual infrastructure and reuse the framework in order to focus in the relevant business logic.
- Equipment manufacturers and solution providers who seek added value through integration and interoperability.

Initially four research communities will benefit from OpenNaaS powered services: the Danish Health Data Network, the British Advance High Quality Media Services, the Irish NGI effort, and NORDUnet through their cloud services These services will be deployed in UNI-C, NORDUnet and HEAnet. The OpenNaaS community

4.3.3 Future of OpenNaas

OpenNaaS has a roadmap, which can be influenced by the community members. A summary of this roadmap is as follows (if you would like to influence this roadmap, please refers to the community section below).

In the short term, a connector for OpenNebula 3.0 will be developed. This will allow the integration of computing and storage resources with network provisioning workflows. While, in a longer term, OpenNaaS aims at being a drop-in replacement for OpenStack's NaaS module also. On the security front, SAML and an ACL engine will allow integration with existing authentication systems (i.e. EduGAIN), while keeping fine-grained access control to OpenNaaS resources.

Currently under development is a porting of the Harmony service [41] , which provides lightweight inter-domain capabilities and Bandwidth on Demand provisioning of transport network services. An infrastructure Marketplace and energy awareness prototypes are also under development. Likewise, OpenFlow support will also be explored.

5. 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.

This section discusses how cloud and infrastructure services are related through the provisioning of interconnected compute and storage resources. Specific attention is paid to how network parameters are provisioned.

5.1 Amazon Web Services (AWS)

Amazon provides both IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) as cloud services. The Amazon IaaS services include the Amazon Elastic Cloud Computing (EC2) and Amazon Simple Storage Service (S3). The following is a short overview of their characteristics and content.

5.1.1 Amazon EC2

Amazon EC2 is a Web service that provides resizable compute capacity in the cloud [42] [43] . The EC2 AMIs (Amazon Machine Instances) forms the basic infrastructure on which applications can be deployed just as any other server. These are available with pre-configured infrastructure resources which can be used for deploying applications and code. Cost of these instances varies based on infrastructure and licenses (open source infrastructure has no such costs), configuration, and type of instance purchased. Instances can be of three types:

- *Reserved* instances, which can be purchased with a one time payment for a long term reservation and are available at considerably lower prices.
- *On-Demand* instances, which are for immediate demands but with comparatively higher prices.
- *Spot* instances, which is unused capacity for which users can bid for.

The AMI allows the following controls over the provisioned infrastructure:

- Network management and access control to the AMI configuration.
- Decide on the location of the AMIs and manage static IP addresses. Currently there are eight availability zones for a user to choose from; US East (Northern Virginia), US West (Oregon), US West (Northern California), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Tokyo), South America (Sao Paulo), and AWS GovCloud.
- Use of a Web based management console. For professional use where this management and creation of instances is required everyday, developers have an option to use AWS command line tools which allows them to write scripts (in scripting language of their choice such as Ruby, bash, python) to automate this management.

To support the AMI, HPC (High Performance Computing) Clusters, which include custom build Cluster Compute and Cluster GPU (Graphical Processing Unit) infrastructure instances, are used to provide high computational and network performance.

In addition, high I/O instances such as virtual machine instances, which use SSD (Solid State Disk) technology to provide I/O rates (> 100,000 IOPS), are used to create high performance NoSQL databases.

5.1.1.1 *Amazon EC2 User Application Component Services*

The following outlines the user application component services offered by Amazon EC2:

- *EBS Elastic Block Store* is a service which provides highly available and reliable storage volumes that are independent and persistent across the life of an AMI. These volumes can

be used as a boot partition for a particular instance or as a storage space with backend replication for long-term durability.

- *Elastic IP Address* is a persistent IP address which can be allocated to a particular user account and provides the user with the ability to dynamically associate / configure them to any AMI. These can even be configured to switch to a replacement instance in the event of a failure in the current AMI.

- *VPC (Virtual Private Cloud)* allows organizations to use AWS resources along with their existing infrastructure in a VPN (Virtual Private Network) to increase compute capabilities beyond the local resources.

- *CloudWatch* is a service to monitor AWS resource utilization, operational performance, and overall demand patterns.

- *Auto Scaling* allows users to dynamically provision and relinquish resources used by their applications depending upon demand in run-time. This can be used to handle sudden spikes in demand without the need to pre-over-provision resources.

- *Elastic Load Balancing* can balance load between multiple AMIs located within a single availability zone or multiple zones. This product can even be used to create fault tolerance where-in the system monitors health of each AMI and distributes load between active instances.

- *VM Import / Export* is a service which provides functionality to upload custom virtual machine images and store a live instance as an image. This feature saves a lot of the effort that goes into the creation of a custom instance (installing infrastructure components, installing applications, security and compliance features etc).

5.1.2 Amazon S3

Amazon S3 (Simple Storage Service) is a service that stores large amounts of data, and is accessible via the Internet [44] . It stores data as objects associated with unique keys. These objects can store up to 5 terabytes of data, and are bound to specific buckets which can only be stored in a particular region (availability zone). This aspect is very important when developing applications which must adhere to data privacy laws that mandate data to be stored in a particular geographical region.

Additional services provided by Amazon include AWS Marketplace that is an online store where users can purchase software to run on AMIs on a pay as you go basis.

5.1.3 Network and IP Addresses Management in AWS

Each EC2 instance, when created is associated with one private and one public IP address. The private IP address is used within the EC2 LAN (Local Area Network). The public IP Address is advertised to the Internet and has a 1:1 NAT (Network Address Translation) mapping to the private IP address of the EC2 instance, and is used to access the AMI by the users over the Internet.

The Elastic IP address is a public IP address accessible on the Internet and is associated with user's EC2 account. A user can associate this IP address to any particular EC2 instance rented with that account, and at any time change the mapping dynamically to a different EC2 instance.

Amazon Route 53 [45] is a DNS (Domain Name Server) service provided by Amazon to map EC2 instance IP addresses to a domain names. Recently this service introduced a new feature, Latency Based Routing (LBR), to route application users to AWS end-points (EC2 instance) which have the best performance (least latency) at the time of request. This can provide the benefit of intelligent DNS based load balancing for applications.

5.2 RackSpace

RackSpace came into existence as an IT hosting company and has gradually expanded to offer IT infrastructure services using the cloud model. The Rackspace Cloud services include the following components:

- *Cloud Files* is a scalable online storage service which provides disk space on a pay for use basis (currently at USD 0.15/GB/month). It can also serve as a CDN (Content Delivery Network) and competes with the Amazon S3 (mentioned above) in the marketplace.

- *Cloud Servers* is an IaaS offering which provides servers with Linux and Windows operating systems to be used for deployment of applications. This service is very similar to Amazon EC2 and competes with EC2 on prices.

- *Cloud Sites* is a PaaS offering which provides an expedient way for businesses to run their software in the cloud. There is a basic monthly fee for a pre-allocation of resources, with additional resources charged on a utility computing basis.

5.2.1 Rackspace General Infrastructure Services

Cloud Sites is a service which provides managed web hosting platform with benefits of cloud technologies like elasticity to handle demand fluctuations.

Hybrid Hosting Solutions is another offering which provides the option of setting up a hybrid cloud within Rackspace's infrastructure. This feature is quite similar to the 'Auto Scaling' feature provided by EC2.

5.2.2 RackSpace Network Service

There is no specific service that allows the user or application to control network performance or topology. Instead the provider declares that they have built an advanced internal network infrastructure with dedicated links and advanced network management infrastructure used only for services provided to customers. This is reflected in their standard Service Level Agreements (SLAs). To use the whole potential of this internal infrastructure, the users need to connect to the RackSpace data centers with the dedicated link.

To provide multiple redundancies in the flow of information to and from the data centers, RackSpace partners with several network providers. To protect from complete service failures caused by a possible network cut, every fiber carrier must enter the data centers at distinct points. The Proactive Network Management methodology monitors route efficiency and end-user performance, automatically improving the network's topology and configuration in real-time. The network's configuration, co-developed with Cisco and Arbor Networks, guards against any single

points of failure at the shared network level. There is an option to extend internal network infrastructure to the customer local/VLAN environment/site. The network is also constantly monitored for latencies and intrusion.

6. Existing Cloud Middleware for Infrastructure Services Provisioning

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

6.1 OpenNebula

OpenNebula is an Open Source Cloud management software [46] that originated from the EC funded project RESERVOIR and is currently is one of the more popular cloud software platforms. The first version 1.0 was release in 2008, and since then, the project is being sponsored by the Distributed Systems Architecture Research Group (http://dsa-research.org) at the Universidad Complutense de Madrid. Subsequently, a new start-up C12G Labs was created to provide professional support for OpenNebula and also private sponsorship of the project. OpenNebula has a strong reputation as an open-source project and is actively used in research and academic communities as a simple and easy tool to setup cloud frameworks [47] .

As a cloud framework, OpenNebula provides users more control over its features, options for customization, and a standard libvirt interface [48] for managing virtual machines. In addition, it uses NFS for storing disk images. OpenNebula has a scheduler which can do the basic tasks of scheduling virtual machines, but its scheduling and lease management can be improved by using another open source lease manager Haizea [49] . Haizea can be used to suspend and resume virtual machines with pre-emptible leases to relinquish resources for higher-priority lease jobs.

OpenNebula as a framework can be deployed on a single machine to host virtual machines or manage other host nodes. A host node in itself is a physical machine which can host virtual machines for the cloud when it is registered with OpenNebula. OpenNebula imposes very little restrictions in terms of configuration and packages that should be deployed on the host machine. Any machine with the libvirt library, the required hypervisor, and a user with name ‘oneadmin’ may be used as a host node.

The OpenNebula core consists of the following components which manage virtual machines, networks, hosts and storage by invoking appropriate drivers (see Figure 6.1.1):

- SQL Pool: is a database which provides the persistence state of the cloud and can be used in the event of a failure to recover the last persisted state.
- Request Manager: consists of an XML-RPC interface which exposes most of the functionality of this framework and is invoked using a command line interface.
- VM Manager: monitors and manages virtual machines using libraries like libvirt.
- Host Manager: manages host nodes registered with an OpenNebula installation.
- VN (Virtual Network) Manager: handles IP and MAC addresses, and provides functionality for the creation of virtual networks.

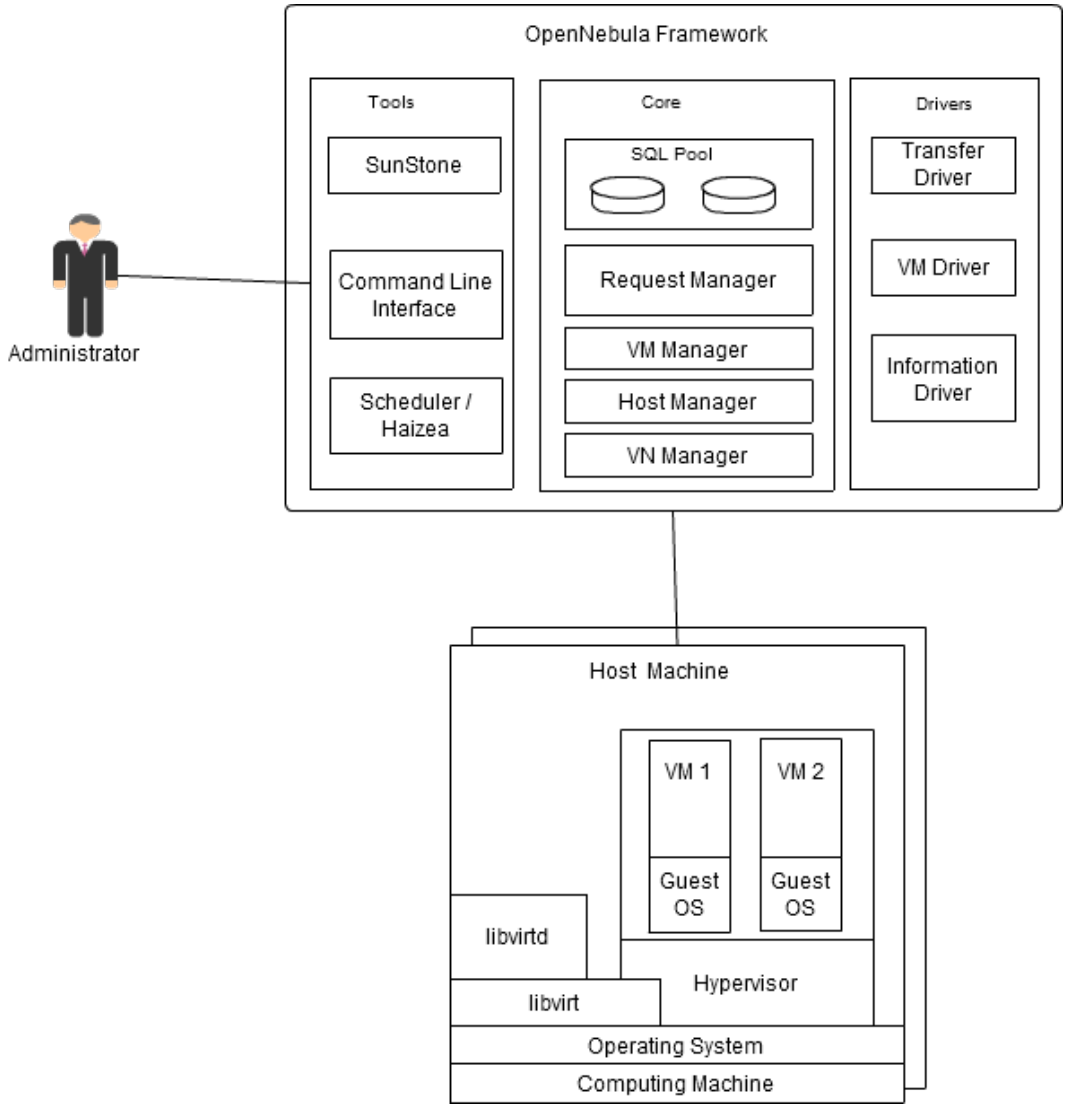

Figure 6.1.1 OpenNebula and its components

Administrative functions allow creating and managing an IaaS infrastructure and include the following tools:

- Sunstone: is a GUI (Graphical User Interface) created in Ruby which can be used to invoke OpenNebula's command line interface to create and manage the cloud.
- Command Line Interface: is an API (Application Programming Interface) which can be directly used to create and manage the cloud.
- Scheduler / Haizea: OpenNebula comes with its own scheduler which can be used to schedule actions within the cloud. OpenNebula also provides option to use Haizea an open source lease manager which can perform scheduling and lease management.

Administrator can use OpenNebula by logging in to head node (via ssh) and using the 'onevm' command to create a virtual machine. This results in the virtual machine image getting copied to a host node with a hypervisor installed. The head node then uses libvirt to manage the host node to

install the virtual machine image. The host node subsequently creates a virtual NIC which is mapped to a particular IP address and MAC address within the virtual network.

6.1.1 Network Management in OpenNebula

OpenNebula manages network connections for its virtual machines with the help of virtual networks. A virtual network consists of a separate set or range of MAC/IP addresses which can be assigned to a new virtual machine instance. This network is associated to a physical network using a network bridge. To simulate EC2 functionality in the OpenNebula framework, an administrator can attach each host machine to both a private network with no Internet connectivity, and a public network with Internet connectivity. This will give each virtual machine instance one public and one private IP address.

6.1.2 Load-balancing in OpenNebula

OpenNebula has no separate component for applying load-balancing on top of an existing configuration like Elastic Load Balancing provided by AWS. To load balance applications deployed in OpenNebula, it has to be implemented separately on virtual machines. An article by C12G Support Team [50] shows how this can be done using the NGinx load balancer to create a hybrid cloud which scales out to utilize Amazon EC2 resources.

6.2 OpenStack

OpenStack is an open source cloud computing framework targeted at cloud providers and large organizations to be used for the creation of public/enterprise clouds [51] OpenStack is a joint development project between NASA and RackSpace Cloud hosting. It was started in July 2010 and its first version 'Austin' was released in Oct 2010. In that respect, OpenStack is the latest open source cloud computing framework with wide support from industry. The OpenStack framework includes the following components (see Figure 6.2.1):

- *Nova [52] or OpenStack Compute* is a cloud controller for the IaaS system that represents global state and mediates interactions with other components. It has seven main components:
  - *API Server* is a web services interface for accepting user requests.
  - *Compute controller* manages compute nodes which host virtual machines.
  - *Object-store or Swift* is a distributed, fault tolerant, and persistent object storage system which has evolved from Rackspace's code for Cloud Files. It provides the same storage functionality as Amazon S3.
  - *Auth manager* is a component which performs authentication and authorization.
  - *Volume controller* manages volumes, which are detachable block devices, that can be attached to a particular virtual machine instance.
  - *Network Controller* manages virtual networks, which includes the creation of bridges and VLANs between compute nodes.
  - *Scheduler* manages the allocation of resources to virtual machines.

- *Glance* is an image database for persisting and retrieving virtual machine images.

- *Keystone* is an identity management service (available as a separate components since OpenStack Folsom release in September 2012) which manages users, roles and permissions.

- *Horizon* is a web application which provides a managment console or dashboard for managing OpenStack (projects, instances, volumes, images, security access etc).

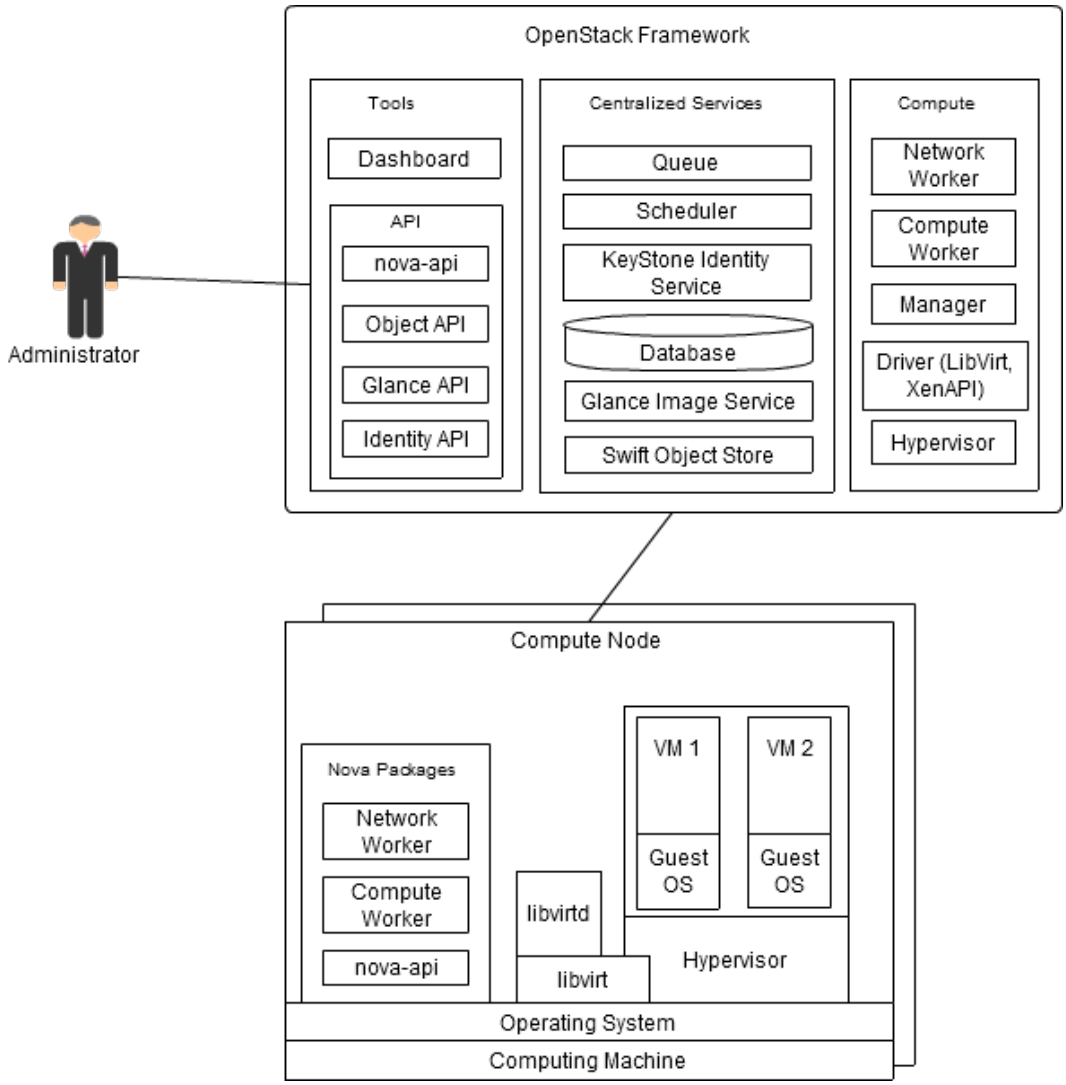

Figure 6.2.1 OpenStack and its components

6.2.1 Network management in OpenStack

The previous version of OpenStack (prior to Folsom release in Sept 2012) network management has been performed by the Network Controller in Nova. Starting from the Folsom release, network management is performed by the independent component Quantum.

Logically Nova supports two types of IP address:

- *Fixed* which are associate with virtual machine instance at creation and remain associated till termination, and
- *Floating* which can be dynamically attached/detached to/from a running virtual machine instance at run-time from Horizon or using the nova-api.

For fixed IPs, Nova supports following three modes of networking:

- *Flat* mode provides each virtual machine instance with a fixed IP associated with a default network bridge. This can be manually configured before an instance is booted. This mode is currently applicable to linux operating systems, which manage network configurations in /etc/network/interfaces (Debian & Ubuntu).
- *Flat DHCP* mode improves upon Flat mode by creating a DHCP server to provide fixed IPs to virtual machine instances.
- *VLAN DHCP* Mode is the default networking mode in which Nova creates a vlan and bridge for each project. Virtual machine instances in the project are allocated a private IP address from range of IPs. This private IP address is accessible only within the vlan. Users can access these instances by using a special VPN instance called 'cloudpipe' which uses a certificate and key to create a VPN (Virtual Private Network).

The Quantum network manager adds the following new functionalities:

- Give cloud tenants an API to build rich networking topologies, and configure advanced network policies in the cloud; e.g. create multi-tier web application topology

- Enable innovation plugins (open and closed source) that introduce advanced network capabilities; e.g. use L2-in-L3 tunneling to avoid VLAN limits, provide end-to-end QoS guarantees, used monitoring protocols like NetFlow.

- allows building advanced network services (open and closed source) that plug into Openstack tenant networks; e.g. VPN-aaS, firewall-aaS, IDS-aaS, data-center-interconnect-aaS.

6.2.2 Load Balancing in OpenStack

Atlas Load Balancer is new component in OpenStack that allows users to apply load balancing to an existing configuration instead of adding a custom implementation for a particular application. It is designed to provide functionality similar to Amazon's ELB (Elastic Load Balancing) and provides a RESTful API for users.

6.2.3 Comparison between OpenNebula and OpenStack

In comparison to OpenNebula, OpenStack is a much larger and more complex framework to understand and install. The basic concepts are still the same, for example the Sunstone GUI (Graphical User Interface) in OpenNebula is functionally equivalent to the Horizon Dashboard GUI in OpenStack. Where OpenStack has the distinct component Glance for handling images, OpenNebula provides that functionality internally. However OpenStack does provides more functionality which are not present in OpenNebula, such as a scalable object store (Swift), and an identity manager (KeyStone).

OpenStack as a framework that can be deployed on a single machine for development and test versions. There is a default shell script named 'devstack', which was initially created by Rackspace and now maintained by the open source community, that contains most of the configuration information. Unlike OpenNebula which can work with any host node with libvirt deployed, Openstack requires each host node to have at least three Nova packages deployed: Network Worker, Compute Worker, and nova-api.

6.3 Eucalyptus

Eucalyptus was developed as an open source replacement for Amazon's EC2 for large enterprises planning to setup their private cloud [53]

Eucalyptus started as a research project in HPC (High Performance Computing) in Computer Science Department of the University of California, Santa Barbara to create an open source framework similar to AWS (which is a closed source platform). Because of its similarity with AWS features, this framework became quite popular and was used and supported by NASA for their research. Since 2010, NASA has shifted its support to OpenStack as an open-source platform of choice. Currently Eucalyptus development is supported by the Eucalyptus Systems, Inc. founded in 2009 to provide commercial support for its framework. In 2012, Eucalyptus Systems, Inc. signed a deal with AWS to collaborate on development and increase interoperability between the Eucalyptus framework and AWS platform.

Figure 6.3.1 illustrates the main Eucalyptus components:

- *Cloud Controller* is the single point of control for the cloud and contains a Web Based interface and Euca2ools command line interface meant to help administrators interact with the cloud and its components. It is used by administrators to manage images, clusters, users, nodes, and infrastructure components within the cloud. It also exposes an AWS compatible interface.

- *Walrus* is storage service for storing large amounts of data generally used for persisting virtual machine images. In comparison to AWS its purpose is similar to Amazon S3.

- *Cluster Controllers* manage each cluster within the private cloud. There is a separate Cluster Controller for each cluster and it is responsible for managing nodes contained within that cluster. Node machines are physical machines with a Node Controller deployed on them to be used as a host machine for creating virtual machine instances.

- *Node Controller* is a part of the Eucalyptus framework used for managing a particular node machine. It interacts with a specific hypervisor (e.g. KVM, Xen, ESX, ESXi, etc) to create and manage virtual machines, acting as an intermediary to receive virtual machine images, and deploying them on nodes.

- *Storage Controller* is a storage component specific to a particular cluster which may be used for storing volumes, taking virtual machine snapshots, and attaching/detaching volumes. It is Eucalyptus' equivalent to Amazon's EBS (Elastic Block Storage).

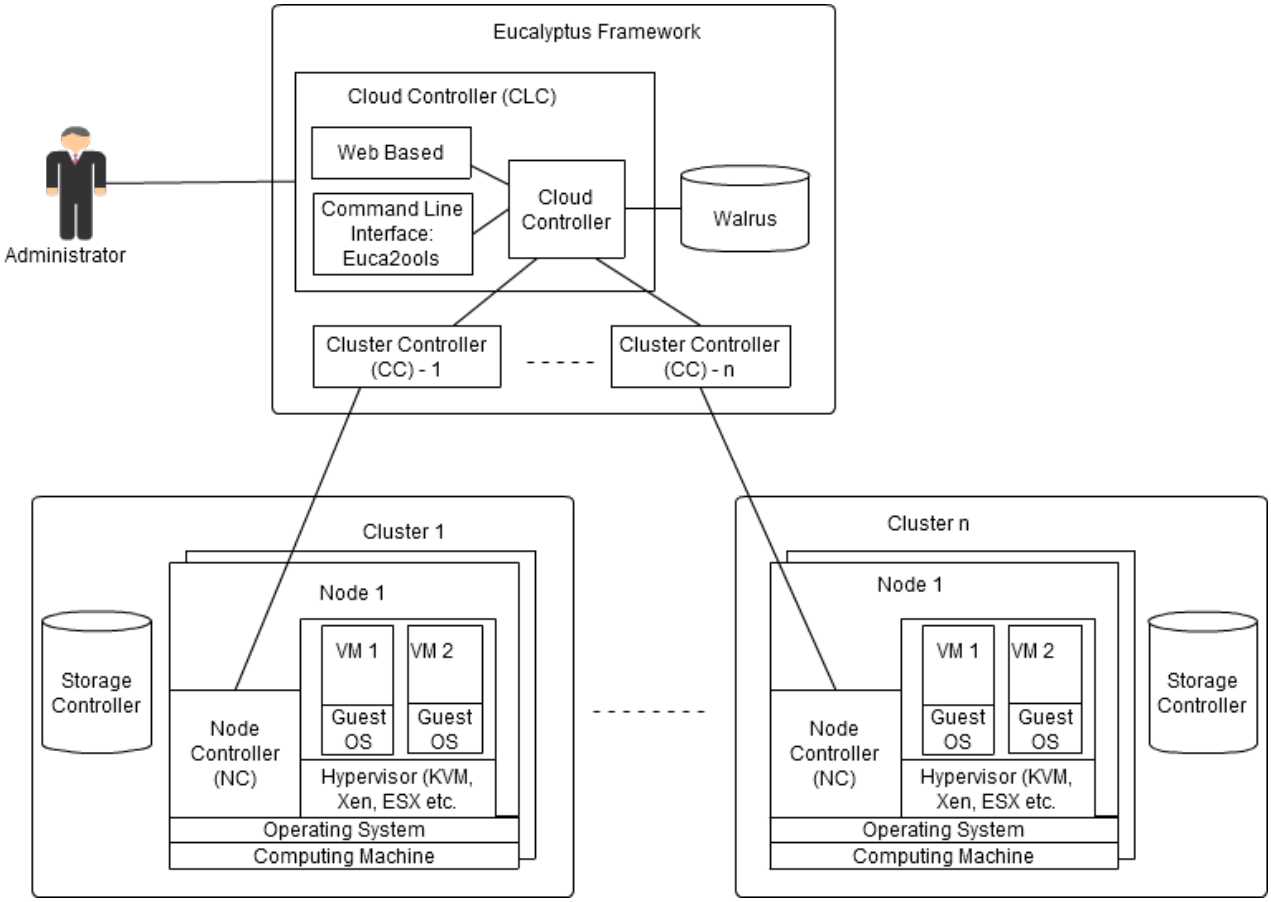

Figure 6.3.1 Main functional components of the Eucalyptus framework

6.3.1 Network Management in Eucalyptus

Networking for virtual machines created by Eucalyptus framework follows similar pattern as EC2. Each virtual machine is given a private IP address and public IP address. The private network is created using VDE (Virtual Distributed Ethernet) which is a virtual network framework to create 'ethernet compliant' virtual networks. A public network is created and configured by cluster controllers which may assign public IP addresses in one of the three ways:

- Using a DHCP (Dynamic Host Configruation Protocol) server
- Selecting IP addressess from a particular range
- Manually configured by an administrator

To provide functionally parity to Amazon's EC2 framework, Eucalyptus supports the concept of Elastic IPs where a user is given control of an IP address to associate it to any one of the running virtual machine instances created by the Eucalyptus framework.

Since version 1.5, Eucalyptus has provided a highly configurable networking subsystem which allows administrators to choose between one of the following four networking modes:

- *System* mode is the simplest but offers the least number of features. Each virtual machine instance is assigned a random MAC address before it is booted. The IP address associated with the MAC address is taken from a DHCP server (which means the DHCP server has to be configured).

- *Static* mode maintains a map of MAC and IP addresses which the administrator can allocate for each virtual machine instance created. In this mode, the framework controls the DHCP server to keep track of IP addresses allocated.
- *Managed* mode provides all the networking features available in Eucalyptus, such as management of security groups, creation of Elastic IPs, and network isolation for virtual machines. In this mode, the network controls the DHCP server and allows users to create 'security groups' or 'named networks' to apply ingress rules to particular protocols and ports. IP addresses are allocated and network ingress rules are applied based on the network to which the virtual machine belongs to.
- *Managed-NOVLAN* mode is similar to Manage mode described above but offers no network isolation.

6.3.2 Load-balancing in Eucalyptus

Eucalyptus currently has no separate component for applying load-balancing apart from the existing configuration like Elastic Load Balancing provided by AWS. However there are indications that the development of such a component is destined for a future release.

7. Existing standards

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

7.1 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.

7.1.1 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

7.1.2 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 7.1.2.1 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 7.1.2.1, 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 7.1.2.1 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 7.1.2.1 Main actors in CCRA and Cloud service provisioning

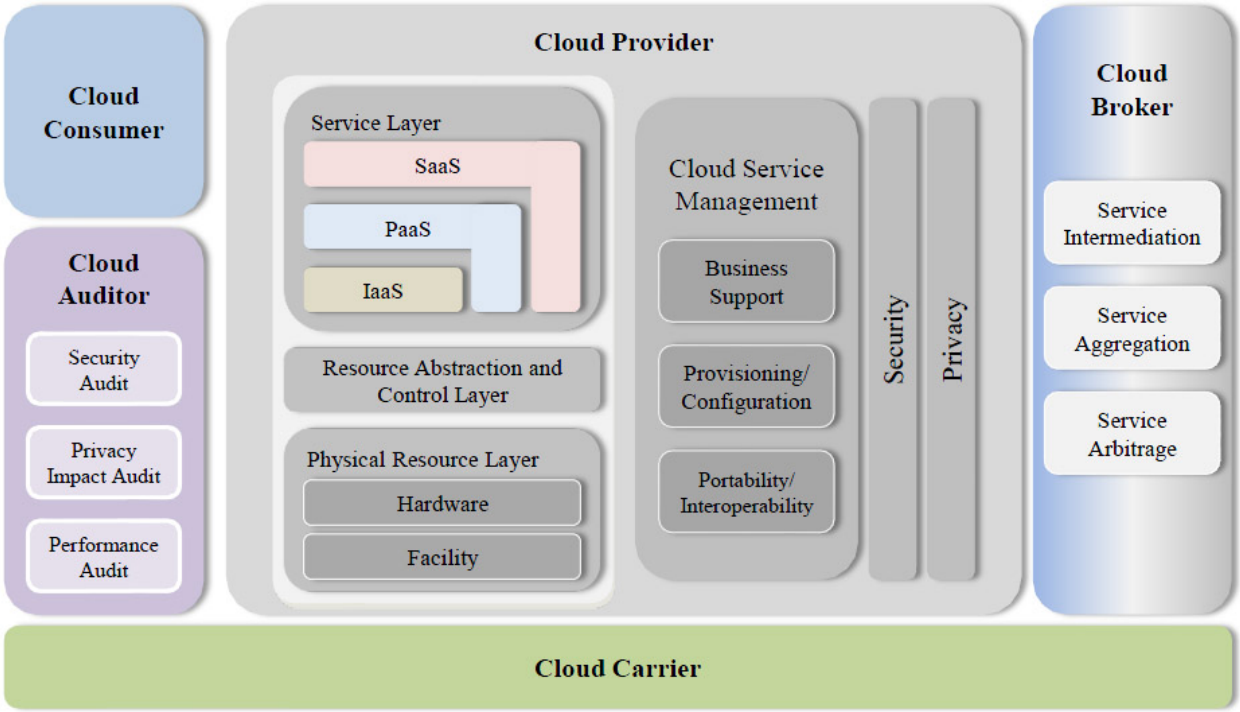

Figure 7.1.2.1 NIST Cloud Computing Reference Architecture (CCRA)

(From: NIST SP 500-292 "Cloud Computing Reference Architecture" document, Sec 2.1, Figure 1, page 10)

Figure 7.1.2.2 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.

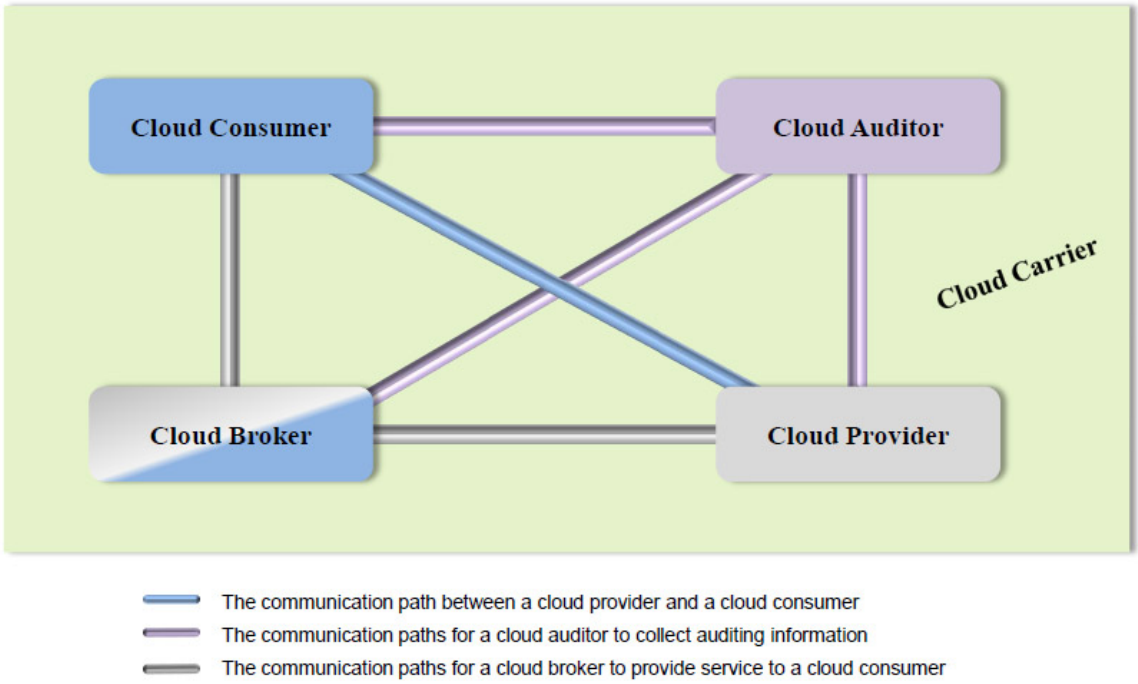

Figure 7.1.2.2 Main role in CCRA service provisioning
(From: NIST SP 500-292 "Cloud Computing Reference Architecture" document, Sec 2.1, Figure 2, page 11)

7.1.2.1 *Relevance and Limitation of the NIST CCRA for Provisioning Network Infrastructure as a part of IaaS.*

The above described 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 the current CCRA inclusion of the Cloud Carrier as representing a typical role played by telecom operators that can provide network connectivity as a 3rd party service, there is no well-defined service model on 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 with the 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 low network latencies.

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.

At a minimum, the following extension/improvement should be made to the cloud IaaS model to meet the wide ranging requirements of 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 virtualization as one of cloud features
- 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.)

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

7.2 IEEE Intercloud Working Group (IEEE P2302)

IEEE P2302 Working Group recently published a draft Standard on Intercloud Interoperability and Federation (SIIF) [54] 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 where all resources and applications are located and run in multiple clouds that are federated in a manner similar to Contend Distribution Networks (CDN). The proposed architecture tries to replicate the CDN approach but does not 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 the extended use of the XMPP as the 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 the Internet approach in building hierarchical or layered interconnected infrastructure for IP based services to support intercloud communication. In actuality, there is no need for the additional intercloud layer to provision and operate cloud based infrastructures because when provisioned, the cloud based infrastructure can function as a normal infrastructure using traditional Internet technologies directly to support intra- or inter-provider communications. This is a viable solution if the appropriate network virtualisation and address translation technologies are configured appropriately. Cloud technologies provide a virtualisation platform for IT and network services to facilitate the instantiation of entire (virtual) infrastructures including related protocols, core services, and control and management functions.

7.3 IETF

The service orientation and utility/infrastructure focus approach of clouds are generally outside the scope of the IETF, however a number of initiatives in IETF have made attempts to address network and protocol related issues in cloud operation and intercloud interoperability. Currently, cloud infrastructure related issues are considered in a number of WGs related to address allocation for distributed data centers, CDN inter-connection (BoF), and incidents reporting and attack protection (BoF).

The Cloud Bar BoF initially took place at IETF78 (August 2010) and has resulted in the submission of several Internet Drafts providing information to the Internet community on the current Cloud Computing standardisation and practices, a proposal for a general cloud and intercloud framework, and a description of a Cloud Service Broker.

7.3.1 Cloud/DataCenter SDO Activities Survey and Analysis

The "*Cloud/DataCenter SDO Activities Survey and Analysis*, *Internet-Draft, version 0.2*, December 28, 2011" [55] document presents a view of current industry involvement in Cloud/DataCenter computing and networking standards efforts, and a prospect of corresponding activities in various cloud standards development organizations (SDO).

7.3.2 Cloud Reference Framework

The "*Cloud Reference Framework. Internet-Draft, version 0.4,* December 27, 2012" [56] document intends to propose a framework for developing new standards to support cloud services through the network, transport, and messaging services. The document defines the Cloud Reference Service Model and Inter-Cloud Architecture Framework. The proposed Cloud Services Reference Model defines a number of horizontal layers:

- *User/Customer Side Services/Functions and Resources Layer(USL)*
- *Access/Delivery Layer (ADL) hosting also Inter-Cloud functions*
- *Cloud Service Layer(CSL)*
- *Resource Control (Composition and Orchestration) Layer(RCL)*
- *Resource Abstract and Virtualization Layer (RAVL)*
- *Physical Resource Layer (PRL)*

and a number of Cross-layer functions including:

- Cloud Management Plane that supports the following functionality

- Configuration management
- Services registry and discovery
- Monitoring, logging, accounting and auditing
- Service Level Agreement (SLA) management (SLAM)
- Security services and infrastructure management

The draft also provides the definition of the Inter-Cloud Architecture Framework (ICAF), which must address the following requirements:

- ICAF should support communication between cloud applications and services belonging to different service layers (vertical integration), between cloud domains and heterogeneous platforms (horizontal integration).
- ICAF should provide a possibility that applications could control infrastructure and related supporting services at different service layers to achieve run-time optimization and required Quality of Service (QoS) (typically related to inter-cloud control and management functions).
- ICAF should support cloud services/infrastructures provisioning on-demand and their lifecycle management, including composition, deployment, operation, and monitoring, involving resources and services from multiple providers (this is typically related to service management and operations support functions).
- Provide a framework for heterogeneous inter-cloud federation.
- Facilitate interoperable and measurable intra-provider infrastructures.
- Explicit/Guaranteed intra- and inter-cloud network infrastructure provisioning (as NaaS service model).
- Support existing Cloud Provider operational and business models and provide a basis for new forms of infrastructure services provisioning and operation.

More specific Inter-cloud functional requirements are articulated as follows:

- Provide a mechanism for resource search and discovery, to determine which serving cloud might have certain resources available (including a match making mechanism)
- Provide a mechanism to authenticate participating entities belonging to different cloud domains.
- 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 multitenancy.
- Provide a mechanism for monitoring, assuring, and troubleshooting across the interconnection.
- Provide a mechanism for defining the monitoring metrics such as Delay-Jitter-Loss. This may be useful for monitoring a flow such as TCP/UDP between IP prefix and a destination address across the interconnection.

The proposed Inter-Cloud Architecture Framework includes the following components with the related interfaces:

(1) Intercloud Control and Management Plane (ICCMP) for intercloud applications/infrastructure control and management, including inter-applications signaling, synchronization and session management, configuration, monitoring, run time infrastructure optimization including VM migration, resources scaling, and jobs/objects routing;

(2) Intercloud Federation Framework (ICFF) to allow independent clouds and related infrastructure components federation of independently managed cloud based infrastructure components belonging to different cloud providers and/or administrative domains; this should support federation at the level of services, business applications, semantics, and namespaces, assuming necessary gateway or federation services;

(3) Intercloud Operation Framework (ICOF) which includes functionalities to support multi-provider infrastructure operation including business workflow, SLA management, accounting. ICOF defines the basic roles, actors and their relations in sense of resources operation, management and ownership. ICOF requires support from and interacts with both ICCMP and ICFF.

The new draft provides a good definition of the network related functionalities in typical cloud infrastructures and has a move in the direction of considering the cloud based infrastructure services as an integrally management component.

7.3.3 Cloud Service Broker

The "*Cloud Service Broker, Internet-Draft, version 0.3*, March 26, 2012" [57] 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 use the CSB to gain access to cloud computing services and resources from Cloud Service Providers. When the CSB receives the cloud service consumer requests, it will select the appropriate cloud computing services and resources from the Cloud Service Providers and determine a specific function pattern to execute cloud service related operations such as intermediation, proxying, monitoring, transformation/portability, governance, provisioning, screening, substitution, security, and composition of services. The CSB will then invoke and adapt to the committed cloud services and resources presented by the various Cloud Service Providers, and return a response to the Cloud Service Requesters/Consumers.

As defined in the Service Orchestration Protocol Network Architecture document [58] , the CSB is equivalent to a proxy, and the specific Cloud Service Providers' platform is similar to a Service Node in the SOP Network Architecture.

7.4 ITU-T Focus Group Cloud Computing

The ITU-T Focus Group on Cloud Computing (FG-Cloud) [59] was established to identify the telecommunication aspects, i.e. the transport via telecommunications networks, security aspects of telecommunications, service requirements, etc., in order to support cloud services/applications and suggest the further studies and ITU-T standardization activities.

As a result of its chartered operation in 2010-2011, the FG-Cloud published the Technical Report (Part 1 to 7) [60] that presents taxonomies, use cases, functional, cloud infrastructure and reference architecture definition, cloud security. The documents also analyse the cloud technology benefits from telecommunication perspectives and discuss scenarios with inter-cloud peering, federation and brokering.

The document "Part 2: Functional requirements and reference architecture" defines the layered Cloud computing architecture that includes the following layers:

- Resources and network layer (including physical resources, pooling and orchestration, pooling and virtualisation)
- Cloud services layer (including basic cloud services IaaS, PaaS, SaaS and also Orchestration service)
- Access layer (including endpoint functions and inter-cloud functions,) where the role of network service providers is defined as to provide inter-cloud transport network
- User layer (including user functions, partner functions, administration functions).

The document "Part 3: Requirements and framework architecture of cloud infrastructure" provides well-defined general requirements to cloud infrastructure from the point of view of the telecom providers. The proposed cloud infrastructure definition are based on the convergence principle recently adopted by telecom industry that uses "one wire" concept for convergence of service traffic, storage traffic, backup traffic, and management traffic.

The document proposes the model for cloud network infrastructure that includes core transport network, intra-cloud network, and intercloud network. Issues related to network interface cards (NIC) virtualisation and virtual machines migration are discussed. The document provides suggestions for cloud network topology design and definition of the virtualised network components such as cloud switch and cloud routes.

7.5 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.

7.5.1 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.

7.5.2 Network Service Interface Working Group (NSI-WG)

The OGF Network Service Interface Working Group (NSI-WG) was formalized in OGF24 (2008). The primary function of the group is to standardize interfaces for network services such as a point-to-point connection service. The larger focus of the group is to formalize an architecture framework whereby atomic network services, such as connections, measurement, monitoring, and protection/restoration could be composed into customized complex services.

To date, the NSI has developed and released an initial connection service interface (OGF NSI CS v1.1) [61] . This protocol has been adopted by several network provisioning systems including AutoBAHN, G-Lambda, OSCARS, OpenDRAC, and OpenNSA/Argia. Interoperability between these provisioning systems using the OGF NSI CS v1.1 protocol has been demonstrated at several venues including GLIF 11 (Rio de Janeiro, Brazil, Sep 2011), FIA (Poznan, Poland, Oct 2011), and SC11 (Seattle Washington, USA, Nov 2011).

7.5.3 Network Markup Language Working Group (NML-WG)

The goal of the NML-WG is to define a schema for description of connection-oriented computer networks. This schema is to be used to exchange topology information between domains for the purpose of path finding, lightpath provisioning, and monitoring.

To date the NML working group has published an informational document describing the state of the art of network description projects and their applications. The NML is currently finishing the final draft of the base schema which will also be used by the new Topology Service currently being defined by the NSI-WG.

8. Summary

This document has attempted to present a cross section of infrastructure provisioning services and practices, ranging from academic to commercial solutions, as well as emerging and current standards that dictate such models. As computing requirements grow and cloud computing matures, *aaS models will gain favor over build-you-own services for a majority of business and academic establishments.

One of the key points of this document is to highlight the requirement for network connectivity to/from/within the cloud to be considered as a “first class” resource, similar to compute and storage. The current view of networks as an infrastructure utility, and not a managed resource, will introduce inefficiencies and unpredictability as the cloud model of computing becomes ubiquitous.

9. 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] *The Open Group: Integrated Information Reference Model* [Online] Available at: http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap22.html

[16] [Online] Available at http://www.sjaaklaan.nl/pivot/entry.php?id=142

[17] *AutoBAHN* [Online] Available at http://autobahn.geant.net

[18] *GÉANT Project* [Online] Available at http://www.geant.net/pages/home.aspx

[19] *GÉANT Bandwidth on Demand (BoD) Service* [Online] Available at http://bod.geant.net

[20] *G-lambda* [Online] Available at http://www.g-lambda.net/

[21] *GridARS* [Online] Available at http://www.g-lambda.net/gridars/.

[22] 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.

[23] 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.

[24] *Fenius* [Online] Available at http://code.google.com/p/fenius/.

[25] *OGF (Open Grid Forum) NSI-WG (Network Service Interface Working Group)* [Online] Available at http://forge.gridforum.org/sf/projects/nsi-wg.

[26] *OSCARS* [Online] Available at http:// http://www.es.net/services/virtual-circuits-oscars/software/configuration/oscars/

[27] 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.

[28] *perfSONAR* [Online] Available at http://www.perfsonar.net

[29] *IDC Protocol Specification v1.1* [Online] Available at http://www.controlplane.net/idcp-v1.1-ns/idc-protocol-specification-v1.1.pdf

[30] *GENI ORCA Control Framework* [Online] Available at http://geni-orca.renci.org.

[31] 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.

[32] 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.

[33] *The Prot´eg´e Ontology Editor and Knowledge Acquisition System* [Online] Available at http:// http://protege.stanford.edu/

[34] *Jena – A Semantic Web Framework for Java* [Online] Available at https://jena.apache.org

[35] *NEuca* [Online] Available at https://geni-orca.renci.org/trac/wiki/NEuca-overview

[36] *Generalised Architecture for Dynamic Infrastructure Services (GEYSERS Project)* [Online] Available at http://www.geysers.eu/

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

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

[39] *Mantychore FP7* [Online] Available at: http://dana.i2cat.net/mantychore-fp7-ip-networks-as-a-service/software/

[40] *OpenNaaS* [Online] Available at: http://www.opennaas.org/

[41] 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

[42] *Amazon web services (n.d) Amazon Elastic Compute Cloud (Amazon EC2)* [Online] Available at: http://aws.amazon.com/ec2/

[43] Watson, M. (2010) *Amazon Web Services A Developer Primer* [Online] Available at: http://www.developer.com/services/article.php/3872986/Amazon-Web-Services-A-Developer-Primer.htm

[44] *Amazon S3 service* [Online] Available at: http://aws.amazon.com/s3/

[45] *Amazon Route 53* [Online] Available at: http://aws.amazon.com/route53/

[46] *OpenNebula 2.0 Architecture* [Online] Available at: http://opennebula.org/documentation:archives:rel2.0:architecture

[47] *OpenNebula Position on OpenStack Announcement* [Online] Available at: http://blog.opennebula.org/?p=683

[48] *libvirt Virtualization API* [Online] Available at: http://libvirt.org

[49] *Haizea: An Open Source virtual machine-based lease management architecture* [Online] Available at: http://haizea.cs.uchicago.edu/

[50] C12G Support Team (2011) *Scaling out Web Servers to Amazon EC2* [Online] Available at: https://support.opennebula.pro/entries/366704-scaling-out-web-servers-toamazon-ec2

[51] *OpenStack Logical Architecture* [Online] Available at: http://docs.openstack.org/trunk/openstack-compute/admin/content/logical-architecture.html

[52] *Nova Concepts and Introduction* [Online] Available at: http://nova.openstack.org/nova.concepts.html

[53] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D. *The Eucalyptus Open-source Cloud-computing System*. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 124–131, 2009

[54] IEEE P2303/D0.2 *Draft Standard for Intercloud Interoperability and Federation (SIIF)*, January 2012 [Online] Available at: https://www.oasis-open.org/committees/download.php/46205/p2302-12-0002-00-DRFT-intercloud-p2302-draft-0-2.pdf

[55] *Cloud/DataCenter SDO Activities Survey and Analysis, version 0.2*, December 28, 2011, [Online] Available at http://tools.ietf.org/id/draft-khasnabish-cloud-sdo-survey-02.txt

[56] *Cloud Reference Framework. Internet-Draft, version 0.4*, December 28, 2012, [Online] Available at: http://www.ietf.org/id/draft-khasnabish-cloud-reference-framework-04.txt

[57] *Cloud Service Broker, Internet-Draft, version 0.3*, March 26, 2012, [Online] Available at: http://tools.ietf.org/id/draft-shao-opsawg-cloud-service-broker-03.txt

[58] *SOP Network Architecture, Internet-Draft, version 0.0*, January 4, 2012, [Online] Available at: https://tools.ietf.org/html/draft-dalela-sop-architecture-00

[59] *ITU-T Focus Group on Cloud Computing* [Online] Available at: http://www.itu.int/en/ITU-T/focusgroups/cloud/Pages/default.aspx

[60] *FG Cloud Technical Report (Part 1 to 7)* [Online] Available at: http://www.itu.int/en/ITU-T/focusgroups/cloud/Documents/FG-coud-technical-report.zip

[61] *OGF NSI Connection Service Protocol v1.1* [Online] Available at https://forge.ogf.org/sf/docman/do/downloadDocument/projects.nsi-wg/docman.root.draft_deliverables/doc16045