OpenNaaS based Management Solution for inter-Data Centers Connectivity

José Ignacio Aznar[1], Manel Jara[1], Adrián Roselló[1], Dave Wilson[2], Sergi Figuerola[1]

[1] *Distributed Applications and Networks Area (DANA), i2cat Foundation, Barcelona, Spain*

[2] *Ireland´s National Education and Research Network (HEAnet), Dublin, Ireland*

Email:{jose.aznar, manel.jara, adrian.rosello, sergi.figuerola}@i2cat.net, dave.wilson@heanet.ie

***Abstract*— Recently, cloud-alike end-to-end services development has evolved towards new service paradigms in which network service provisioning management among Data Centers represent a key enabler. In this paper, we address the issue of network service provisioning management among Data Centers. We propose a software based management solution leveraging virtualization technologies, applied on top of network and Data Center infrastructures. Our solution offers a Network as a Service (NaaS) based service which relies on two main open source software components: OpenNaaS and OpenStack. We present the design in an already-working testbed that demonstrates the utility of the management mechanism.**

Keywords—NaaS, OpenNaaS, OpenStack, management, test-bed, virtualization, cloud, IT, datacenter, DC.

I. INTRODUCTION

The sheer increase of cloud-based solutions and business models embracing the IT world is driving new Data Center (DC) architectures, moving from a static set of equipment vertically dedicated to different applications, towards more dynamic and geographically distributed pools of shared IT resources. Such dynamic infrastructures reveal new requirements in network virtualization, service delivery automation and resource management that should be supported by well-defined information models and related middleware while providing inter-cloud provisioning solutions.

Inter-DC connectivity provisioning services is not a novel topic; nevertheless, users' cloud computing and storage needs together with the increase of traffic exchanged across network providers' infrastructures becomes critical for an efficient delivery of Real-time online interactive application (ROIA) [1] while coordinating IT and network infrastructure services. Several limitations have been observed so far: for instance, IT service availability and continuity management plans demand geographically distributed backup resources for recovery purposes. However network provisioning services among customers and the DCs that host such services are restricted to static connectivity solutions based on fixed recovery paths. Also, an inter-DC end-to-end (E2E) network provisioning service would have to be able to traverse through several network segments potentially managed by different network providers. This fact pushes the boundaries to network provisioning automation and management since coordination of all providers is required to establish the inter-DC E2E path. Moreover, the crossed domains might comprise several technologies (e.g. optical ROADM-based domains, MPLS domains, carrier Ethernet campus, etc.) which imply limitations in both data (non-compatible interfaces) and control (protocol limitations, control information hidden to other domains) planes. Consequently, technology specific constraints limit path provisioning services and E2E management too.

Thus, inter-DC E2E provisioning services pose considerable challenges from the management perspective as the range of providers, network technologies and lack of automation limits a Data Center's versatility while providing IT and cloud based services. All these limitations invite one to consider a solution based on a common management framework, capable of providing E2E inter-DC connectivity services in a flexible and reliable way, with guaranteed QoS, by leveraging the most up-to-date IT and network tools and technologies. This enables an end-to-end integration of networks and IT community, which overcomes previous described limitations.

In this paper, we address the issue of network service provisioning management among Data Centers. To this end, we propose a software based management solution, leveraging virtualization technologies applied on top of network and DC infrastructures. The solution offers a Network as a Service (NaaS) which relies on two main open source software components: OpenNaaS [2] and OpenStack [3] , [22]. OpenStack is a software designed to deploy and manage large networks of virtual machines. It is composed of several inter-related modules that coordinate to provision computing, storage and networking services to a cloud administrator. OpenNaaS is a management platform that enables the abstraction of underlying network technologies and offers NaaS-based services. It provides the capability to configure and deploy their own services, enabling administrators to manage any part of the infrastructure or to deploy any type of application on top. In our solution, OpenNaaS delegates such management to OpenStack, so that all the E2E inter-DC provisioning service is managed by the cloud manager.

We propose the design of the system architecture and we present such a design in an already-working testbed as part of the Mantychore FP7 [4] project. Finally we provide a use case example that demonstrates the utility of the management mechanism, leveraging OpenNaaS and OpenStack virtualization technologies.

The reminder of this paper is organized as follows. Section II describes the state of art in terms of network and cloud management tools. It also goes through other testbed initiatives which comprise the joint management of IT and network for provisioning purposes. Section III describes the system architecture, the testbed implementation and the main benefits of its performance. Section IV presents a use case that demonstrates the framework utility. Section V concludes the paper and depicts open research lines to enhance the mechanism.

II. Related Work

Related technologies in terms of software platforms:

A. State of art in terms of Cloud software technologies:

Although there were many other different solutions [19], we take into consideration two major initiatives as open source cloud management software platforms: OpenNebula and OpenStack. The main criteria to choose these platforms were the large community behind these two projects, that they present new features and functions constantly and that they both have the possibility of creating extensions on top of them.

OpenNebula [5] is an open source software toolkit with the aim to offer a standard solution for building and managing large distributed infrastructures: private, public and hybrid clouds. OpenStack is a popular open source cloud computing platform for public and private clouds. This platform is further described in the following section, and was the cloud management system chosen for the implementation of the service. The factors taking into account to choose OpenStack were the larger number of hypervisors supported and the increase in the number of participants in the community [6], in addition to having a network component called Quantum[1] with an accessible open API.

B. Network related software platform and frameworks:

Nuage Networks [7] provides software defined networking solutions for datacenter networks. Their Virtualized Services Platform comprises of three products: Services Controller, Services Directory and Routing & Switching. They offer a virtualization environment for a datacenter network, but these software-based products are not open source and were discarded because of the difficulties on using and expanding an implementation based on these products in the future.

Tail-f Network Control System [8] provides a single interface to manage all network devices, application and services. As well as Nuage Networks product it is designed to be a generic SDN solution and it is also a commercial product.

C. BoD Provisioning systems

Automated Bandwidth Allocation across Heterogeneous Networks (AutoBAHN) [9], [21] is the only service currently existing outside the research and education community for provisioning bandwidth-on-demand [20]. AutoBAHN is a GÉANT tool which automatically finds the best path across different networks and integrates with an NREN's own systems to facilitate the creation of dynamic circuits in multi-domain environments.

D. Previous/similar solutions or testbeds:

Also some recent initiatives have developed per-layer optimization solutions, in which on-demand network services between predetermined Cloud infrastructures are provided. Thus, the FP6 PHOSPHORUS project [10], [24] worked on a solution for the coordinated provision of network and grid resources; nevertheless, its performance resulted in an explosion of control messages. More recently, the FP7-GEYSERS [11], [23] project has proposed an architecture capable of provisioning Optical Network and IT resources for end-to-end service delivery based on virtualization and partitioning mechanisms. The test-bed implemented for demonstration purposes was specific to optical network technologies.

III. System Architecture

The system architecture is based on three main different software platforms: OpenStack as cloud manager, OpenNaaS as the network provider for OpenStack and AutoBAHN as the BoD provisioning tool.

A. OpenStack modules

OpenStack is a collection of open source technologies delivering a massively scalable cloud operating system. Several components work together in order to provision computing, storage and networking services to a cloud administrator. Table 1 shows the list of modules used in the solution design that we propose with a brief description.

TABLE I. OpenStack projects and components [12]

Project (module)	Component	Description
Nova	Compute device	Virtual servers and volumes
Glance	Image service	VM disk images
Quantum	Networks	Secure virtual networks
Cinder	Block storage	Volumes management
Keystone	Identity	Authentication and Authorization
Horizon	GUI	OpenStack GUI interface

- Glance [13]: Virtual machines built by OpenStack boot from virtual disk images. Glance is a shared component that manages them and offers a catalog repository of available images.
- Cinder [14]: Cinder provides block storage to guest virtual machines. It may be located either at the same physical host as the VMs or in a remote host.
- Nova [15]: Nova provides virtual servers on demand. Nova provides a scalable compute platform, supporting

[1] OpenStack Networking name was called Quantum and changed to Neutron during the implementation of the service

the provisioning and management of large numbers of servers and virtual machines.

- Quantum [16]: Quantum offers networking as a service to the rest of the OpenStack components. It's developed with a pluggable architecture to support many different networking solutions. Using the available REST proxy plugin in the testbed, OpenStack is able to communicate with a remote application which is able to provision network as a service, i.e. OpenNaaS.
- Keystone [17]: This is a shared service that provides authentication and authorization for all the OpenStack services. It manages access to compute servers and images in Glance for example.
- Horizon [18]: Horizon is the canonical implementation of OpenStack's Dashboard, which provides a user interface to OpenStack services including Nova, Swift, Keystone, etc.

In our design, OpenStack services manage Data centers' connectivity and also triggers requests to OpenNaaS for network provisioning.

B. *OpenNaas Framework*

OpenNaaS is a platform for provisioning network resources that allows the deployment of dynamic network infrastructures by network operators, supporting many different types of application, each with their own access and network resource usage patterns.

This platform offers a robust and extensible Network as a Service (NaaS) open source framework where network virtualization can be achieved. The NaaS paradigm is the foundation on which OpenNaaS works as a key enabler to address this challenge by offering virtual infrastructures to third party through this platform (by means of web services API). OpenNaaS allows the creation of a virtual representation of a physical resource (e.g. network, router, switch, optical device or computing server), based on an abstract model of that which is often achieved by partitioning (aka slicing) and/or aggregation.

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 network operators can rely on the continuity, transparency and adaptability of the platform.

The OpenNaaS framework is characterized by the resources which present manageable units inside the NaaS concept. A Resource can be (among other things) a switch, a router, a link, a logical router or a network, and each resource is composed of one or more capabilities. A capability is an interface to a given resource functionality (e.g. OSPF configuration in router resource). By modeling underlying the network in terms of resources and capabilities, OpenNaaS is capable to abstract such network to get information from it and trigger configuration actions, on top of it.

OpenNaaS tool can be divided in three layers, as can be seen in (Fig. 1): Platform, NaaS layer and Network intelligence. The platform is the base of the application. It provides the application context and defines its technological framework, including communication channels to access OpenNaaS. In this layer, common components based on the NaaS abstraction are defined. Controllers for such components are also provided. This platform serves as the basis for the NaaS layer.

The NaaS layer reuses platform defined components to expose user manageable resources. This layer offers support for concrete resource types, functionalities (called capabilities) and device types. Resources and capabilities are exposed to the user abstracted from implementation details and from hardware specifications. Hence, this layer acts as a Hardware Abstraction Layer.

Applications willing to manage the network may be built on top of the abstraction provided by the NaaS layer. These applications compose the Network Intelligence layer, as it is here where managing policies may be defined and actions based on them applied to the network using OpenNaaS. Normally, these applications are not shipped with the OpenNaaS distribution, but independently from it.

OpenNaaS, triggered from OpenStack, is in charge of configuring the connectivity of the network domains segments.

Among all the resources and plugins or capabilities developed for OpenNaaS, it is of special interest for this scenario the Quantum resource and the interface to AutoBAHN. From that point, all other underlying resources and functionalities of OpenNaaS are exposed and can be used by OpenStack.

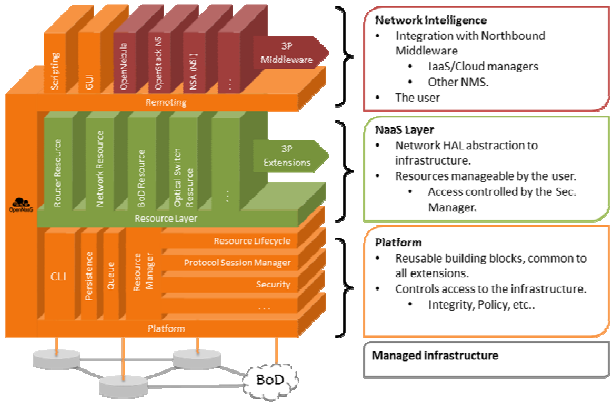

Fig. 1. OpenNaaS architecture.

C. *AutoBAHN*

Automated Bandwidth Allocation across Heterogeneous Networks (AutoBAHN) is a GÉANT bandwidth-on-demand provisioning tool that allows users to provision multi-domain circuits through a web interface (or API) in few minutes. It was the tool chosen for the project Mantychore FP7 as the

Bandwidth on Demand (BoD) service and it is integrated with OpenNaaS.

D. Architecture Design

The system architecture (Fig. 2) enables cloud administrators to be provided with a management solution to coordinate both cloud and network management while creating inter-DC E2E connections, based on OpenStack and OpenNaaS tools. The system makes use of OpenStack interface (CLI, web service or the Horizon [18] OpenStack GUI) to access the management platform and request E2E connectivity services. OpenStack is in charge of configuring the DCs´ resources and network and it is also the responsible of granting the rights to provide the service among the DCs involved in the service. For network operations, OpenStack triggers requests to OpenNaaS for network provisioning through the Quantum REST Proxy plugin.

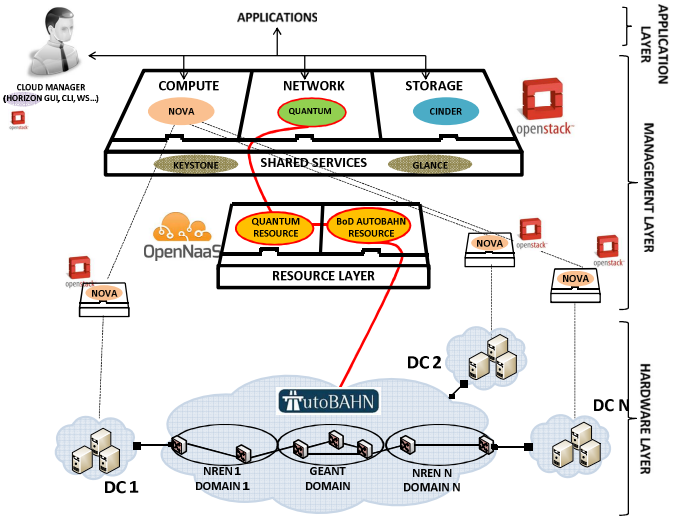

Fig. 2. System architecture.

The Quantum plugin implements two important functionalities: a data model and a RESTful web service client. The Quantum data model consists of three main entities: network, sub-network and port. A network is an isolated virtual layer-2 broadcast domain. Over this network, the administrator can create different sub-networks. Each sub-network consists of a block of IP addresses that define a Layer-3 domain. The RESTful web service client communicates with an external network provider; in our case, OpenNaaS. The Quantum plugin, together with OpenNaaS, offers CRUD (create, read, update and delete) methods for the previous three entities, so that network, sub-network and port can be created, managed and deleted on-demand with the RESTful service. More specifically, all "Read" methods are directly sent to the plugin database, while the remaining methods are delegated to OpenNaaS before updating the information in the plugin database.

In this specific solution, OpenNaaS makes use of one of the implemented resources in the Mantychore project: the "quantumresource" is used by OpenNaaS to access some data that the network provisioning request coming from OpenStack does not contain. Therefore, the resource model has been developed based on the Quantum model specification.

The quantum resource in OpenNaaS implements the "QuantumAPIV2" capability, which exposes a RESTful web-service to communicate the "quantumresource" to the Quantum Plugin. This way, CRUD methods over network and ports are delegated to the network provider by means of the Quantum plugin.

The "QuantumAPIV2" capability contains two important components: the network builder and the model controller. The network builder is responsible for mapping quantum information into requests to other OpenNaaS resources. In our testbed, the "QuantumAPIV2" capability maps the OpenStack requirements (basically the end points to be connected) and instantiates the AutoBAHN resource. AutoBAHN is an automated bandwidth-on-demand provisioning tool that provides the best path between two layer 2 endpoints across different networks, to create a circuit between them. This way, the OpenNaaS platform communicates with the AutoBAHN resource (also developed in OpenNaaS) and requests the creation of a layer 2 network between the endpoints specified in the OpenStack request. It's important to note that the testbed is built on top of existing NREN and GEANT facilities; where AutoBAHN is available, the testbed has been able to use this from the beginning. AutoBAHN intelligently provides the layer 2 network provisioning service. That said, OpenNaaS has been developed in a flexible way, so that another provisioning tool can be used to satisfy users' needs.

The information of how the new network is mapped in OpenNaaS is stored by the model controller. As its name indicates, the goal of this component is to update the information of the quantum resource model, not only with the data of the network built by the network builder, but also with the information sent by the Quantum plugin. Fig. 3 shows the detail of the interconnection between OpenStack and OpenNaaS and the different modules and capabilities that shape the system.

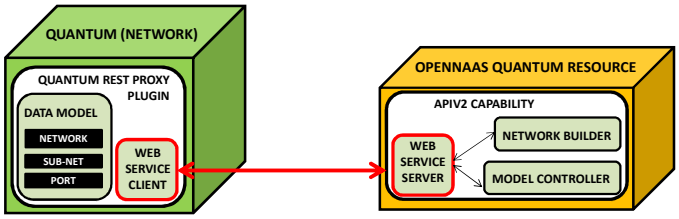

Fig. 3. Detail of the Quantum plugin and relevant system modules.

All in all, OpenStack manages both a cloud and network E2E inter-DC provisioning service. OpenStack makes a request to OpenNaaS to establish the network between DCs, and then later instantiates a provisioning tool (AutoBAHN) to provide the E2E connectivity. This way, the complete solution provides an E2E on-demand provisioning service.

E. Testbed implementation

In our demonstration scenario, OpenStack components were divided into two different hosts. The first one, called

Malakk, is located in the HEAnet offices (Dublin). It contains all OpenStack components, acting as the controller node of this OpenStack instance, but also as a host for virtual servers. The second one, installed in the DeiC offices (Copenhagen), hosts the OpenStack NOVA component and additional virtual machines. The third server of the scenario hosts the OpenNaaS platform, which is listening for any Quantum request that needs the support of network provisioning. For that purpose, OpenNaaS exposes a RESTful Web Service (WS) implemented following the Quantum Plugin API specifications.

The testbed comprises two separate and independent networks between hosts. The first one (black color connections in Fig. 4) is the control network, and is used to allow the controller node to communicate with all the Nova compute nodes. Since Malakk and DeiC hosts don't reside in the same network domain, the Internet connection is used to send and receive control flows to the compute nodes, such as VM creation requests. Therefore, both hosts should be accessible through the internet and they should know each other's IP address in the initial scenario. The second network (red color connections in Fig. 4) is built on-demand by OpenNaaS. The purpose of this network is to act as the data network between the virtual machines of both hosts. From the several available ways to perform the inter-DC connection with a remote host within the same network, we chose to create a Layer 2 link on demand, integrating AutoBAHN, OpenStack and OpenNaaS in the same scenario. Even though OpenNaaS supports other valid solutions for this use case, like GRE, one additional motivation of creating a L2 link was to take advantage of all capabilities provided by the networking service of OpenStack (Quantum), as the DHCP server. When creating a virtual machine, the DHCP agent assigns to it a new IP address from the available pool of addresses for this network. This IP address is reserved until the virtual machine boots and sends the DHCP request. Since DHCP requests are sent to the local broadcast domain, we need the DHCP server to be placed in the same network domain as the virtual machines; we achieve this by linking both hosts with a L2 network. DHCP requests and responses travel through the AutoBAHN created circuit from one host to another, and virtual machines are able to get their IP addresses automatically.

In order to use an AutoBAHN circuit to interconnect both hosts, the network interface connected to the AutoBAHN Enabled Switch should be configured with the VLAN used in the AutoBAHN entry port, as part of the initial scenario as well (VLAN 100 in Fig. 4, for instance). Since OpenNaaS does not interact with physical hosts, the VLAN configuration has to be set manually. Development of OpenStack software was outside the goals of the Mantychore project. Therefore, to automate this process in the future, we suggest developing a quantum L2 agent by which the Quantum plugin could create network sub-interfaces and assign them a single VLAN.

OpenNaaS will use these VLANs to request a virtual circuit in AutoBAHN network. Once it's built, a L2 network is available between both hosts. The cloud administrator will use OpenStack to create a subnet; that is, a pool of IP addresses that could be attached to a L2 network. All virtual machines created in either host will receive an available IP from this pool of addresses, and will create a private L3 network distributed between the remote hosts.

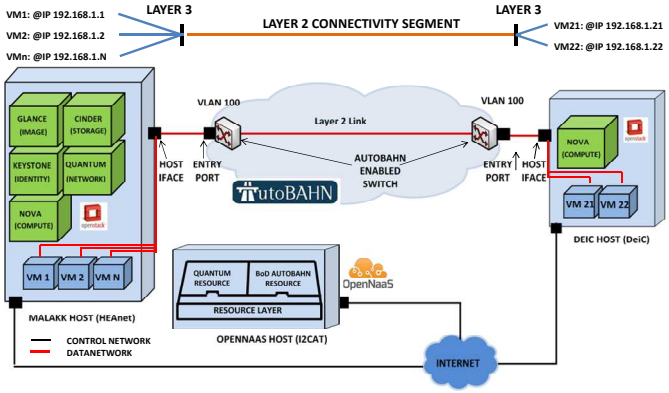

Fig. 4. Test-bed implementation at HEAnet, DeiC and I2CAT facilities.

F. *Key benefits*

There are several benefits that any institution obtains with the implementation of that service. First of all, it allows expanding DCs easily by creating, duplicating or migrating VMs on-demand. The interconnection of public and private DC makes this service very useful for any contingency plan; where backups and dynamic redundancy can be fully automated and institutions can have their services duplicated in different geographical locations. Companies can also offer some real-time critical services from different locations, closer to the end user, minimizing the delays. That results in a service for network operators and cloud service providers, where they can manage their DCs with more flexible and using a reliable toolset.

IV. A PRACTICAL USE CASE

In this section we present a use case built on top of the testbed previously described. It has been developed for the Mantychore FP7 project and has been presented as an internal demo. This use case is the starting point of a service that can be used (or offered) by some partners of the Mantychore project, obtaining the benefits mentioned in the previous section. The use case main goal is to enable given cloud infrastructure provider to extend his or her Data Center by connecting two remote clusters using OpenNaaS as the tool provisioning the network between them, mixing the IaaS and the NaaS paradigms in a single scenario.

In our scenario (workflow described in Fig. 5), the storage capacity of a certain enterprise is about to run out and a new capacity plan specifies the need to extend the DC by creating new virtual machines at a remote DC. The cloud manager is the role in charge of such DC extension by means of the solution we propose in this paper.

Thus, the cloud manager accesses the OpenStack Horizon interface (1) from which she will manage the E2E connectivity service creation in two sequential steps. Firstly, she requests the creation of the network segment between the DCs. To this target, she must specify the network segment endpoints (i.e. the Data Center demarcation points) and the range of IP

addresses that correspond to the VMs to be connected (2). The Quantum plugin, as explained in section III, forwards this information to OpenNaaS (3) which will provide the NaaS-based connectivity service. OpenNaaS specifies the VLAN and ports to be interconnected with each other to the AutoBAHN BoD provisioning tool in order to create the E2E circuit (4). Both ports and VLANs currently are statically pre-defined for implementation simplicity. Once the circuit is created, this is announced to the cloud manager (5). The second process step consists of the creation of the VMs. The cloud manager instructs to OpenStack to create them, specifying the network to which they have to be attached, (6) and communicates to OpenNaaS the VMs that have been created and the port to which they are attached (7). Once OpenNaaS has matched the VMs to their network, it confirms the E2E provisioning service towards the cloud manager finalizing the service provisioning (8). This way, the cloud manager is capable of configuring an end-to-end connectivity service by using both network and cloud software tools in a coordinated way.

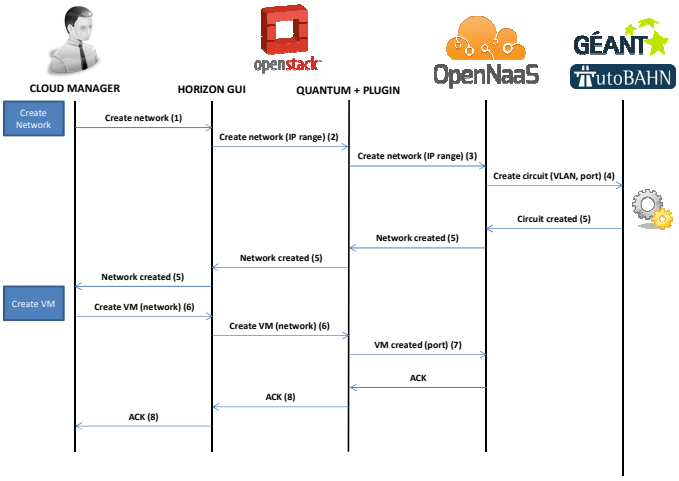

Fig. 5. Use case flow diagram.

V. Conclusions and Future Work

As was previously noted, inter-DC E2E provisioning solutions are not a novel topic. Nevertheless the process automation leveraging the coordination of the most up-to-date virtualization tools is novel. The proposed solution enables an administrator to provide E2E provisioning services gluing together both network and cloud management platforms, and managing both together. Thus, ROIA, cloud-based and NaaS-based services can be offered in an automated and flexible way overcoming current coordination limitations and providing a more flexible performance than previous static network provisioning service approaches.

A further research and test-bed enhancement expects to improve the tool automation. For instance, in the current version of the platform, it is required machines hosting VMs to be directly connected to the AutoBAHN switches. It is planned to enrich the tool while including DC networks segment within the E2E provisioning service. Another planned enhancement consists of the automation of the VLANs assignment while provisioning the network segment, since now, they are statically pre-configured. The idea is to assign a range of VLANs from which the software tool is capable to make the assignation process. Also interesting is the idea to enable OpenNaaS with failure recovery capabilities in case certain network resources fail (instead of just notifying such failures).

Acknowledgment

The work in this paper has been funded by the European Union through the project MANTYCHORE (FP7-261527).

References

[1] L. Dong, “ROIA: The Architecture and State Consistency, ” the 4th International Conference on Multimedia Information Networking and Security (MINES), Japan, November 2012.

[2] http://www.opennaas.org/

[3] http://www.openstack.org/

[4] http://www.mantychore.eu/

[5] http://opennebula.org/

[6] X. Wen et Al, “Comparison of Open-Source Cloud Management Platforms: OpenStack and OpenNebula, ” the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012), China, May 2012.

[7] http://www.nuagenetworks.net/

[8] http://www.tail-f.com/network-control-system/

[9] http://geant3.archive.geant.net/service/autobahn/about_autoBAHN/Pages/About_autoBAHN.aspx

[10] PHOSPORUS FP6 project (2009), *White Paper on Optical Networking Requirements for European NRENs.*

[11] http://www.geysers.eu/

[12] M. T. Jones, “Cloud Computing and Storage with OpenStack, “ http://www.ibm.com/developerworks/cloud/library/cl-openstack-cloud/

[13] http://wiki.openstack.org/wiki/Glance

[14] http://wiki.openstack.org/wiki/Cinder

[15] http://wiki.openstack.org/wiki/Nova

[16] http://wiki.openstack.org/wiki/Quantum

[17] http://wiki.openstack.org/wiki/Keystone

[18] http://wiki.openstack.org/wiki/Horizon

[19] Gregor von Laszewski et Al, “Comparison of Multiple Cloud Frameworks, ” the IEEE Fifth International Conference on Cloud Computing, Honolulu (USA), June 2012.

[20] http://geant3.archive.geant.net/service/BoD/about_BoD/Pages/home.aspx

[21] Campanella M. et Al, “Bandwidth on Demand Services for European Research and Education Networks, “ the IEEE 1st International Workshop on Bandwidth on Demand, San Francisco (CA), November 2006.

[22] Sefraoui O. et Al, “OpenStack: Toward an Open-Source Solution for Cloud Computing, “ International Journal of Computer Applications, Vol. 55 – No. 03, October 2012.

[23] Tzanakakit A., et Al, “Virtual Infrastrucutre Planning: The GEYSERS Approach, “in proc. of FNMS 2012.

[24] Figuerola S. et Al, "PHOSPHORUS: Single-Step On-Demand Services Across Multi-Domain Networks for E-Science", Proc. SPIE, vol. 6784, 2007.