Deliverable D13.2 (DJ2.2.1) Specialised Applications' Support Utilising OpenNaaS/NaaS

Deliverable D13.2 (DJ2.2.1)

Contractual Date:	30-12-2014
Actual Date:	09-04-2015
Grant Agreement No.:	605243
Activity:	JRA2
Task Item:	T2
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	RedIRIS/i2CAT
Document Code:	GN3PLUS14-1233-29
Authors:	José I. Aznar Baranda, Eduard Escalona, Sergi Figuerola (RedIRIS/i2CAT), Victor Reijs, Dave Wilson (HEAnet), Pavle Vuletic (AMRES), Ole Frendved Hansen (NORDUnet/DeIC/UNI-C), Freek Dijkstra, Erik Ruiter (Surfnet/SARA), Hao Zhu (Surfnet/UvA), Mohamed Morsey (Surfnet/UvA), Bernhard Schmidt (DFN/LRZ), Jordi Ortiz, Francisco Ros, Antonio F. Skarmeta, Pedro Martínez-Juliá (RedIRIS/UM), Sonja Filipovska (MARnet), Guy Roberts (GEANT Limited).

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract

The GÉANT Network and the NRENs have the infrastructure capabilities and the expertise to offer advanced, special-purpose networking services beyond the 'long fat pipes'. Together with Software-Defined Networking (SDN) and Network (Function) Virtualisation (NFV), Network-as-a-Service (NaaS) provides the means to deliver the next generation Research and Education Network services and operations. "How exactly does this work?", "Where do NRENs stand today?" and "What is the future landscape?" are key questions that JRA2T2 partners address.

To reach this target. JRA2T2 focuses on shaping innovative network service offerings with wide applicability and added-value as a NaaS-based model of management and provisioning for GÉANT end users.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The GÉANT network and the NRENs have the infrastructure capabilities and the expertise to offer advanced, special-purpose networking services beyond the 'long fat pipes'. Together with Software-Defined Networking (SDN) and Network (Function) Virtualisation (NFV), Network-as-a-Service (NaaS) provides the means to deliver the next generation Research and Education Networking services and operations. "How exactly does this work?", "Where do we stand today" and "What is the future networking landscape?" are key questions addressed as part of the work carried out in Task 2.

This document provides an overall perspective on the NaaS service model and the related technological landscape of SDN, NFV, Management as a Service (MaaS) and information modelling. Starting from a set of use cases provided by NRENs, the key NaaS achievements delivered by JRA2 T2 as part of its remit to shape network services for the R&E communities and end users of GÉANT services are presented. Several demonstrators and proofs of concept have been implemented and developed to show the potential of NaaS.

The OpenNaaS software platform has been chosen to implement NaaS services within GN3plus. Thanks to the support of the OpenNaaS community and the feedback received by JRA2T2 partners, a re-thinking of the architecture into something more flexible, easy to deploy and easy to integrate with other control and management platforms has been designed.

This document provides next steps for NaaS, identified risks as well as key recommendations and conclusions.

1 Introduction

The JRA2T2 team focuses on shaping innovative network service offerings with wide applicability and added-value. Potential for a new management and provisioning model for the GÉANT-NREN and user communities, based upon the Network as a Service (NaaS) concept is now being investigated.

JRA2T2 is currently researching NaaS as an architectural and service delivery model for NRENs to use the network to manage and provide network services. This is evaluated within the context of network management, network resource abstraction and virtualisation, and integrated with Network Function Virtualisation (NFV) and Software Defined Networking (SDN). Such network services can be easily extensible according to the user requirements, and provide smart interfaces for advanced performance, control and management.

The JRA2T2 research plan started from a basic NaaS definition, a set of use cases and specific technologies, and a collection of research areas of relevance (see [MJ221]). Several challenges were addressed:

- Enhancement of NaaS use cases and proposals for new ones, adding value to network services, in order to demonstrate the advantages of NaaS principles.
- Turning interesting features into NaaS capabilities has been challenging due to the early-stage development of relevant technologies.
- Creating greater awareness of NaaS services. Disseminating the potential and value of NaaS capabilities and overcoming the slow process of technological acquisition.

JRA2T2 has brought the OpenNaaS framework into GÉANT for prototyping and proving of NaaS concepts. OpenNaaS is as an Open Source platform for GÉANT, NRENs and other infrastructure providers of network and computing resources, enabling NaaS-based services. Within GN3plus, several framework shortcomings, such as a lack of flexibility, were addressed in close collaboration with the OpenNaaS community.

This document is structured as follows:

- **Section 2** provides an overview of NaaS-related initiatives.
- **Section 3** details the NaaS service model, presents outcomes from current research areas and the proposed developments in the GÉANT network.
- **Section 4** demonstrates how OpenNaaS can be used as a tool for tailoring NaaS-based network services to user needs.
- **Section 5** includes the coordination and dissemination activities carried out by JRA2T2 partners.
- **Section 6** summarises achievements, including:
    - NaaS (Network as a Service) definition: Demonstrate how the NaaS model enables the delivery, management and orchestration of network services and applications. Show how OpenNaaS impacts network and service provisioning models.

- Identification, development, implementation and deployment of use cases created to drive analysis, propose service Proof of Concept (PoCs), and demonstrate the value of NaaS.
- Research on technologies of relevance and interest for the community.
- A NaaS vision and roadmap within the GÉANT project and the NREN community, 'pushing the envelope' in terms of next-generation network services.

Two appendices provide more detailed information about the technical achievements of the activity:

- **Appendix A** presents a number of use cases.
- **Appendix B** describes the PoC implemented and deployed to validate the use cases.
- **Appendix C** provides a brief status update of OpenNaaS software development.

Finally, it is worth mentioning that at the time of publication of this document, some of the activities of JRA2T2 are ongoing. The team works on technology integration/inter-operation with NaaS and some of the PoCs will incorporate new features, extending the capabilities presented here.

2 NaaS-Related Initiatives

The user community requirements for advanced network services are reflected in the plurality of commercial initiatives and research projects aimed at automated and flexible network provisioning mechanisms. In most of the cases, research is driven by the requirements to interconnect heterogeneous network technologies and accommodate ever-increasing bandwidth requirements in a cost-effective manner. Differing types of requirements are being addressed in various ways through proposing services, tools, platforms or frameworks. In the following sections, a high-level overview of the NaaS state of the art initiatives prior to the work done in JRA2T2 is presented.

2.1 Cisco Network Service Orchestrator

The Cisco Network Service Orchestrator (NSO), enabled by Tail-f (previously known as Tail-f's Network Control System (NCS)), is a commercial offering from Cisco [NSO]. NCO's objective is to enable network operators to manage all of their services and network devices with a single tool. NSO has many similarities to OpenNaaS, targeting routers, switches and their interfaces.

The NETCONF protocol is used towards the devices or in some cases, (such as NSO) as a technology proxy. NSO is designed for single domain use and does not natively support multiple tenants. NORDUnet is using NSO to configure Juniper routers for network peering (BGP) and VPN.

2.2 Internet2 Initiative OESS

Internet2's Open Exchange Software Suite (OESS) provides on-demand setup of circuits. The service recognises individual users as well as groups, the entities to which endpoints are delegated. A web interface presents relevant endpoints to the user and offers the ability to setup a connection.

"OESS is a set of software used to configure and control dynamic Layer 2 virtual circuit (VLAN) networks on OpenFlow-enabled switches. OESS provides sub-second circuit provisioning, automatic circuit failover, per-interface permissions, and automatic per-VLAN statistics. It includes a simple and user-friendly web-based user interface as well as a web services API." [OESS]

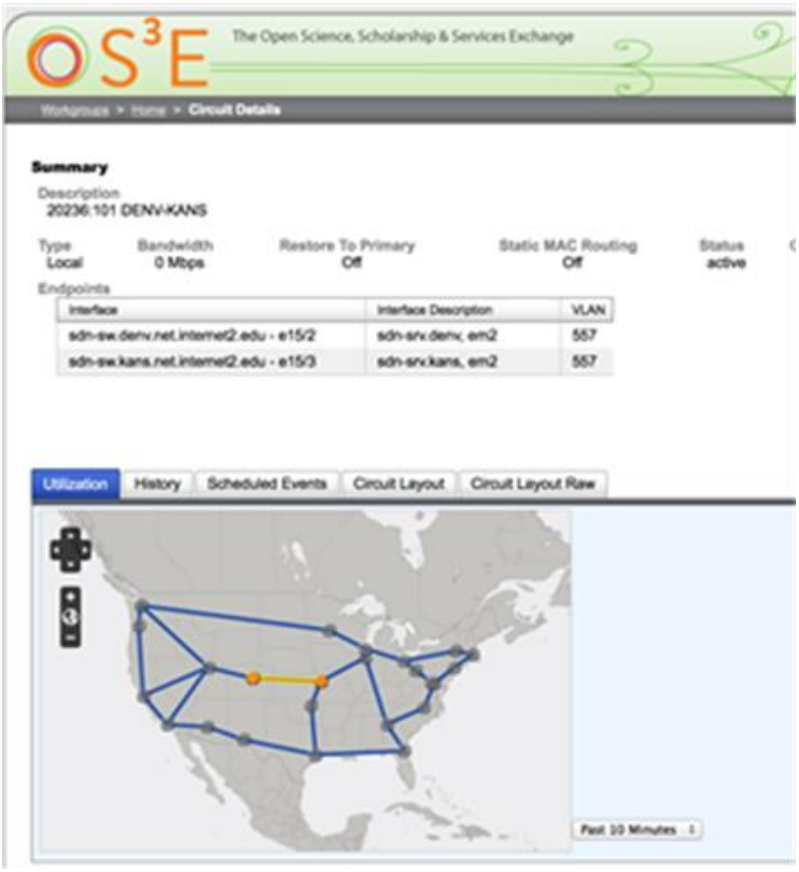

Figure 2.1: OCSS user interface. The user selects a backup circuit through a clickable map

2.3 Proprietary Provisioning Tools

HEAnet's point-to-point Ethernet network (known internally as "Bluenet") is driven by a tool that was developed originally by GRNET and adapted by HEAnet to provision point-to-points.

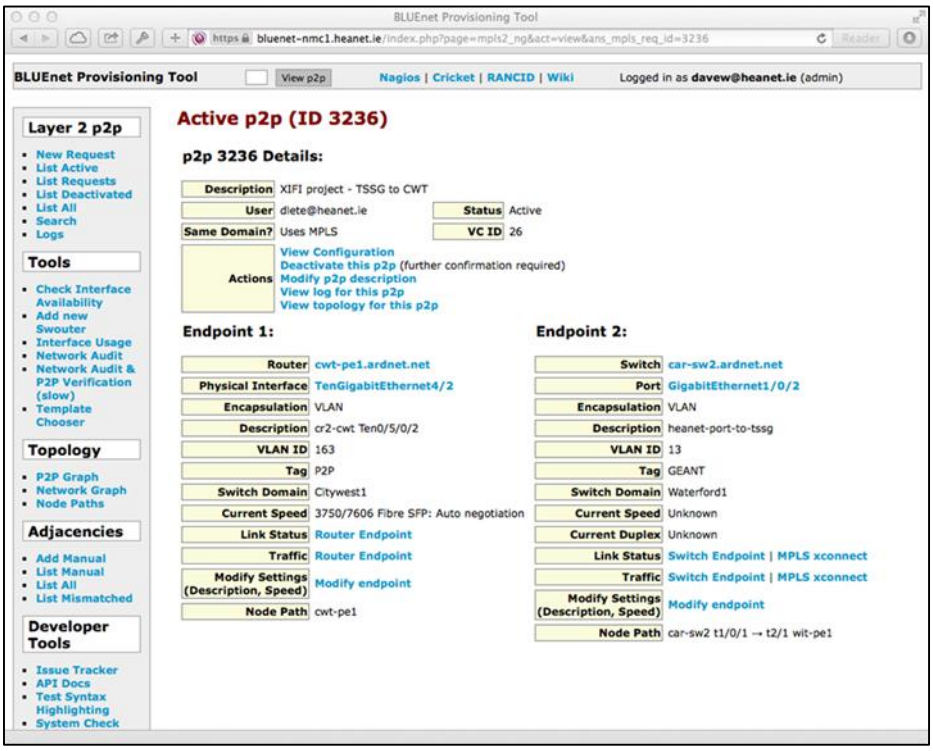

Figure 2.2: Screenshot of Bluenet tool

The network underneath Bluenet is a hybrid of an MPLS-based core around Ireland and switched Ethernet spurs and rings in regional metropolitan areas. The tool, therefore, needs to maintain a topology of the switch and router ports that are available for use, compute the paths between them upon receiving a service request, and provision the combination of switched Ethernet VLANs and possible MPLS circuits needed to fulfil the request. Bluenet also automatically configures monitoring tools so that traffic data is measured.

The tool forms a fundamental part of HEAnet's service to its customers. It has been used to create thousands of point-to-point circuits. It is fast and efficient enough to enable setups that might otherwise not be worthwhile, and it is reliable enough that it requires only the very lightest of change control.

More recently, an API was developed that allows the tool to service requests from GÉANT's Bandwidth on Demand (BoD) system (see below), in effect, turning Bluenet into the Domain Manager for HEAnet. This is a very powerful approach because the complexity of path computing is left with GÉANT, while the segment to be implemented in HEAnet gains the same status (and the same setup and monitoring) as a request from an engineer in the HEAnet NOC.

2.4 GÉANT BoD

The GÉANT Bandwidth on Demand Service is based upon a Layer 2 inter-domain provisioning system controllable by the end user, and as such, is an application of NaaS technology. It works by leveraging the existing networks in NRENs – and in some cases, their existing provisioning systems, such as the Bluenet example described above.

BoD is a production service in GÉANT, although production status varies between NRENs. For instance, SURFnet has crafted a tailored system for its part of BoD. Users are identified through SURFconext, the NL-federated ID, and are given membership of certain teams (groups of users). Teams are provided with tokens needed to enable circuit endpoints. Furthermore, automatic configuring of the last part of the network from the BoD endpoint to the user's equipment through a specific VLAN is possible. This functionality is similar to the required delegation of rights to resources in NaaS. The configuration of the "last mile" is also similar to the Network on Demand use case presented herein.

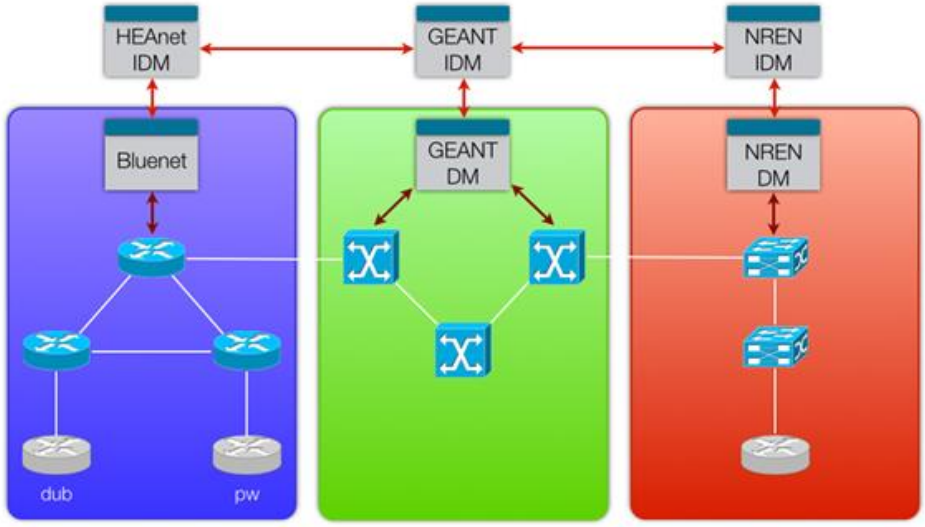

Figure 2.3: Overview on BoD service

2.5 SDN-Based Approaches: SDNv1 and SDNv2

Software Defined Networking (SDN) is complementary to Network as a Service. In fact, one of the ultimate aims of NaaS development is SDN integration, in order to leverage the SDN functions as part of service delivery to users via NaaS. SDN is about separation of control plane intelligence from data plane and abstraction of the control plane. According to the Open Networking Foundation (ONF) definition, SDN is the "the physical separation of the network control plane from the forwarding plane, where a control plane controls several devices." [SDN-ONF]. At the same time, NaaS is about creating network services, managing and orchestrating them and giving control to the user.

Recently, additional definitions of SDN have surfaced, proposing an evolution of SDN towards SDNv2. As discussed in [Shenker], SDNv2 is just emerging as an idea proposed by the author (one of developers of SDNv1). Shenker sees the over-emphasis on Data Centre environment as a reason to look at SDNv2. The SDNv2 proposal provides a better mapping than the SDNv1's implementation (OpenFlow) on the definition of SDN, with regard to redefining abstractions: network control abstractions which are programmable; forwarding abstraction; state distribution abstraction (programs can operate on network graphs); and specification abstraction.

SDNv2 is also looking at including NFV and higher layers (getting closer to the principle of delivering capabilities 'as-a-Service') and targeting legacy equipment/carrier networks. The following principles proposed by SDNv2 include:

- Software goes to the edge; the core stays dumb – although partly relevant to NaaS, it can be a useful design principle for networks and services.
- 'Middleboxes' (Layer 4-7 appliances, real or virtual, found all over the network) are included in SDN. Here, greater relevance to NaaS emerges. On top of control technologies (SDN or any other), NaaS proposes to deploy, configure, orchestrate and chain services. Such services leverage (a) the underlying network infrastructure (either physical or virtualised) and (b) the control features that SDN may expose to the upper layers to make use of the network resources.
- The network is opened up to third-party services – NaaS enables the creation and orchestration of services. It may also open interfaces to integrate with third-party services.
- Closed interfaces are not allowed – NaaS does not prescribe this, as it views existing services/resources as valid components of a service.

There have also been some criticisms of SDNv2, see [SDNv2C].

2.6 Positioning in Relation to Other Initiatives

In JRA2T2, the NaaS model is positioned at the forefront of network provisioning services to accommodate current and future control and management requirements. The recent evolution of virtualisation techniques and infrastructure services not only results in new network requirements but can also provide variable, dynamic and flexible network connectivity services. Thus, NaaS should be seen as a solid opportunity for R&E networks to rethink the established approach to networking, improve its performance and reduce management complexity.

3 NaaS Service Delivery Model

This section provides an overview of the underpinning of NaaS and the approach that it is being followed in JRA2T2 to exploit NaaS in the context of GÉANT, grounded on the showcases, the research areas and a dissemination plan.

3.1 Fundamentals of NaaS

NaaS is a business and service delivery model related to network infrastructure for providing network services with dynamic and scalable, yet secure and isolated, access to networks for multiple tenants. NaaS enables flexible deployment and operation of customised network services, delivery of reliable performance to overlay applications and delegation of management rights to users. The NaaS model has the potential to increase the performance of tenants, while reducing service deployment and management development complexity. The following characteristics make NaaS a smart model for service management and provisioning.

3.1.1 Abstracted and Virtualised Operational Model

The NaaS model proposes decoupling control and management features from the physical topology and vendor-specific details of a network infrastructure so that the former can be offered to the tenants. The model aims to be flexible enough to procure different designs and orientations, but also specific enough so that common tools can be built and reused across plugins. Virtualisation techniques are a key driver for supporting the Network Hardware Abstraction Layer (Net-HAL) and exert a very strong impact on NaaS-based services' development. In fact, abstraction, resource sharing and isolation constitute the basis of virtualisation for NaaS, providing application layers and tenants with the vision of "the network is a resource composed of capabilities that can be controlled and managed".

The fundamental unit used to accomplish this is the Resource. A Resource models a device and represents a manageable unit inside the NaaS concept (e.g., switch, a router, a link, a logical router, a network). The basic resource considered is the Physical Resource (PR); Virtual Resources (VRs) are then created by manipulating the PRs (different paradigms and Virtual Infrastructures (VI) are compositions of VRs into a single, manageable construct). Capabilities shape resource functionalities and provide an interface to given resource functionalities (e.g., for a router: OSPF, IPv6, Create/manage logical routers, etc.). Figure 3.1 shows this NaaS vision in which physical devices are abstracted into resources and capabilities, the management of which can be delegated to application layers.

In this way, as Infrastructure as a Service sets the basis for business models related to virtual machines, NaaS sets the basis for business models related to virtualised network infrastructures.

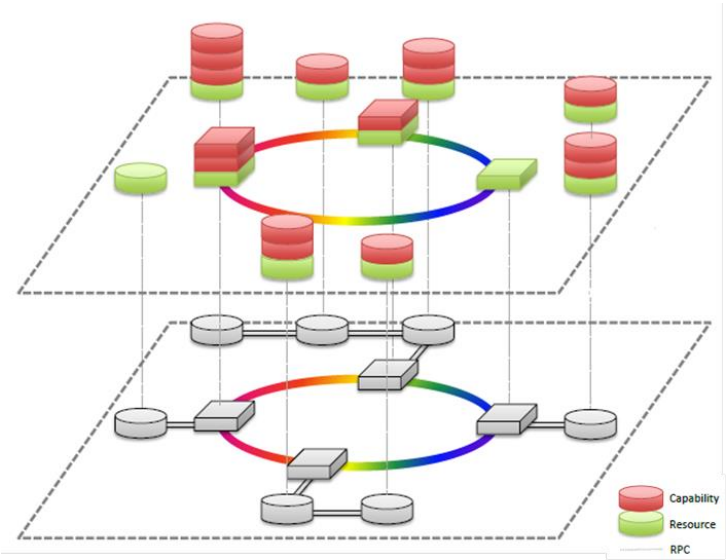

Figure 3.1: NaaS Lightweight abstract view. Resources and capabilities

3.1.2 Decoupled and Coordinated Network Control and Management

Abstraction opens a new landscape for network control and management. Two keywords for the NaaS service delivery model are "*decoupled*" and "*coordinated*." NaaS enables a network model in which functions are distinctively differentiated, establishing clear competences in each of the layers and enabling new roles for SPs and network operators. Such roles and competences are coordinated along different strata and segments so that each of the roles acquired at any level of the infrastructure stack is clearly identified. Figure 3.2 shows the NaaS decoupling points. The separation, shown vertically in Figure 3.2, allows us to robustly share the underlying resource among the different entities involved.

- The Management plane abstracts control functions for human interaction.
- The Control Plane includes signalling among resources for protocol configuration.
- The Transport Plane includes a plethora of network technologies.
- Service and application planes retrieve information data and exert their action over the previous planes by means of connectors (e.g. REST APIs, GUI, etc).

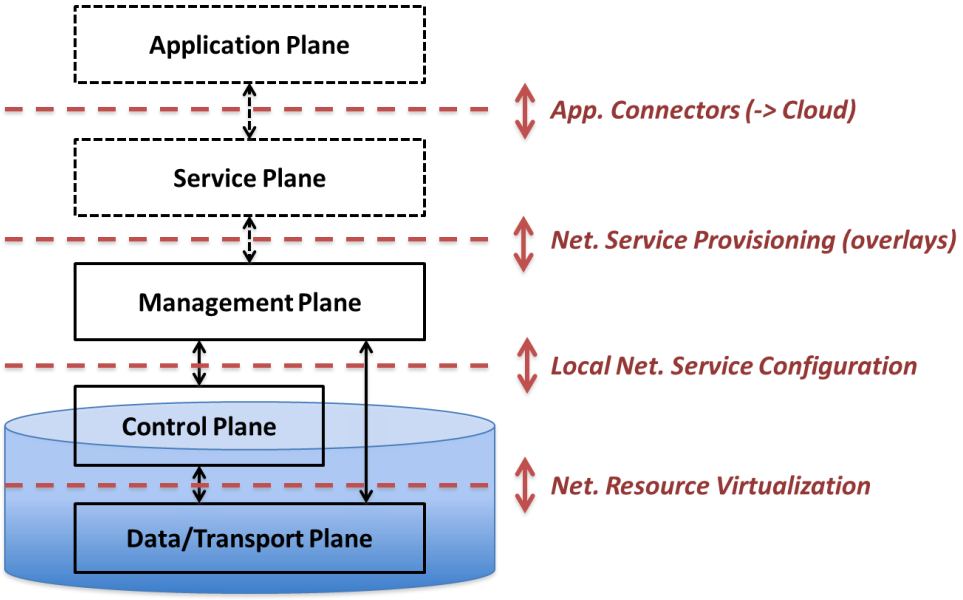

Figure 3.2: NaaS Layer decoupling and coordination

3.1.3 Policed Resource and Capabilities Access, Depending on Access Rights and Ownership

Another NaaS feature is the delegation of access rights, recursively, over the managed resources and capabilities. Ownership and rights are used to define usage policies. The paradigm of ownership specifies what type of actions may be performed over specific resources and who is entitled to trigger them. Because the delegation is recursive, access can be further delegated from one user to another, which is essential in certain environments. For example, an NREN may delegate resources to a university's IT department, who may then sub-delegate some of those resources to a researcher, without requiring a direct relationship between the NREN and the researcher. Based on [Dijkstra] there are four major defined types of ownership considered for the NaaS model:

- Legal or Economic is the entity that purchased the device. It determines its policy of the resource, and carries the responsibility for the behaviour of the device. It has the final responsibility in case of hazards or abuse.
- Administrative ownership to perform management operations over the resource.
- Operational ownership to perform configuration operations over the resource.
- Usage ownership entitles its holder to place, transmit or compute its data into the resource, but strictly prohibits any kind of resource alteration.

Table 3.1 shows a basic set of resource operations as a function of the ownership scheme

	Economic Ownership	Admin. Ownership	Operative Ownership	Usage Ownership
Physical Resource (PR)	Add/Remove	Partition, Management operations	Control operations (through the VR)	Use functionalities offered by the corresponding resource
Virtual Resource (VR)	Create/Remove	Resize, duplicate, snapshot	Control operations (start/Stop for IT VR, Create XC for Net VR)	
Virtual Infrastructure (VI)	Create/Tear-down	Duplicate, snapshot, re-plan	Batched operations through VRs	

Table 3.1: Resource and ownership mapping table

These key features provide the NaaS model with a clear vision: all involved stakeholders are no longer tied to the so-called "network services" in which end-users request for a certain service, and conditioned to service providers' management, control and service provisioning. Instead, while offering NaaS, the control and management is offered as a service to end-users, becoming independent of service providers to build their applications on top of the network according to their specific requirements.

3.2 NaaS in NRENs/GÉANT Community

The NaaS concept represents an interesting service model from which the GÉANT community in general, and NRENs in particular (as service providers), can benefit. NRENs typically provide connectivity services to universities, secondary schools, research institutions or public administration organisations. Such connectivity services are generally based on relatively static IP networking (in terms of Internet access, routing, addressing, etc.). The role of the NREN in these cases is as a (Network) Service Provider.

Over the years, the NREN community has engaged with provisioning tools and virtualisation of networking resources. This has happened trough several EC funded projects over the course of the TEN-155 MBS (Management Bandwidth Service), GÉANT2, GÉANT3 and others. The Bandwidth on Demand project has spanned a number of iterations of the GÉANT project, and the GN3 project included research into a virtualisation service across multiple layers called GENUS. In addition to these, the FP7 project Mantychore (and its forerunners Mantychore I and II), focused specifically on deploying IP Networks as a Service among virtual research communities [Mantychore]. The NaaS management and service delivery model is a very valuable tool that can bring GÉANT services closer to the users, enabling them with control and management capabilities, and at the same time making service delivery more flexible and smarter from an operational point of view.

During GN3plus, the JRA2 Task 2 has carried out a number of actions to extend the NaaS model and guarantee a proper understanding of the technical and business costs, benefits and perceived business value that the adoption of the NaaS model may bring to GÉANT community while provisioning network services. The OpenNaaS software tool constitutes a framework to develop such services. These actions have been mainly driven to:

- Provide **use cases** and derive tangible **showcases** which highlight NaaS model features. The showcases aim to offer specific solutions to overcome concrete limitations in network management and provisioning encountered by GÉANT partners.

- Research on **NaaS-related areas and technologies** to be integrated within the NaaS model. Identification and analysis of the requirements that such technologies impose in the NaaS model.
- Prepare a **dissemination plan** to further develop NaaS model and use cases, and attract the attention from target audience.

3.2.1 Use Cases

In order to bring NaaS from concept to reality, a set of use cases have been proposed. These form a foundation upon which network services may be built to enhance service operations and delivery. The use cases are not limited to the core network segments, but also apply to the access network, the last mile and IT networking environments. In [MJ221] a first approach to these use cases was proposed. More details are provided in Section 5, which describes specific requirements that NaaS services should implement now and in future.

3.2.2 Showcases – NaaS Services

The main target of the task is to provide NaaS based services in the form of showcases, focusing on the enhancement of specific network aspects. Starting from the reference use cases and also upon some additional ones, we have developed several demonstrators and Proofs of Concept (PoCs) which aim to show the added value these network services may bring to the GÉANT community. A full description of the showcases is provided in Section 6. Section 7.2 contains a list of the conferences and events where they have been presented.

3.2.3 Dissemination Plan

During the JRA2T2 team's face-to-face meeting that took place in April 2014, the need to establish a dissemination plan to increase awareness of the NaaS model and attract the attention of GÉANT stakeholders was highlighted. The dissemination efforts continue until the end of GN3plus project, and include the following activities:

- **Dissemination plan calendar:** A calendar of network and service-related events, conferences and magazines has now been set up in which the workgroup can explain the NaaS model and show their outcomes in form of demonstrators and showcases.
- **YouTube videos** of the different demos we have performed at different events. In these videos are some specific NaaS service demonstrators and their associated features based on the OpenNaaS open source software platform [YouTube-demos].
- **NaaS and OpenNaaS training**: During the last months of the project, a NaaS and OpenNaaS webinar will be offered in collaboration with NA1T7. The details are not yet available, but it may consist of two parts: the first to provide an overview of NaaS and the second more technical and developer-oriented to enable attendees to play with OpenNaaS coding some specific simple services. The target is to engage GÉANT partners either as users/consumers of NaaS services created on top of OpenNaaS, or as developers to create their own services on top of the OpenNaaS core.

3.2.4 Research Areas – Requirements for NaaS-Based Services

Together with the use cases, the NaaS-related research guides and informs this task. Research studies enable analysis of the impact that NaaS service capabilities have on a number of topics related to the NREN community. The research started at the beginning of the task aims to link MaaS, NFV, SDN and information modelling advancements to the NaaS service model. Figure 3.3 shows the variety of researched areas: the bigger the bubble, the more interest it has received from partners. The hottest topics at the moment are network virtualisation techniques, SDN and NaaS interconnection and NFV environment positioning. In Section 3.4 key achievements from each of these areas are presented.

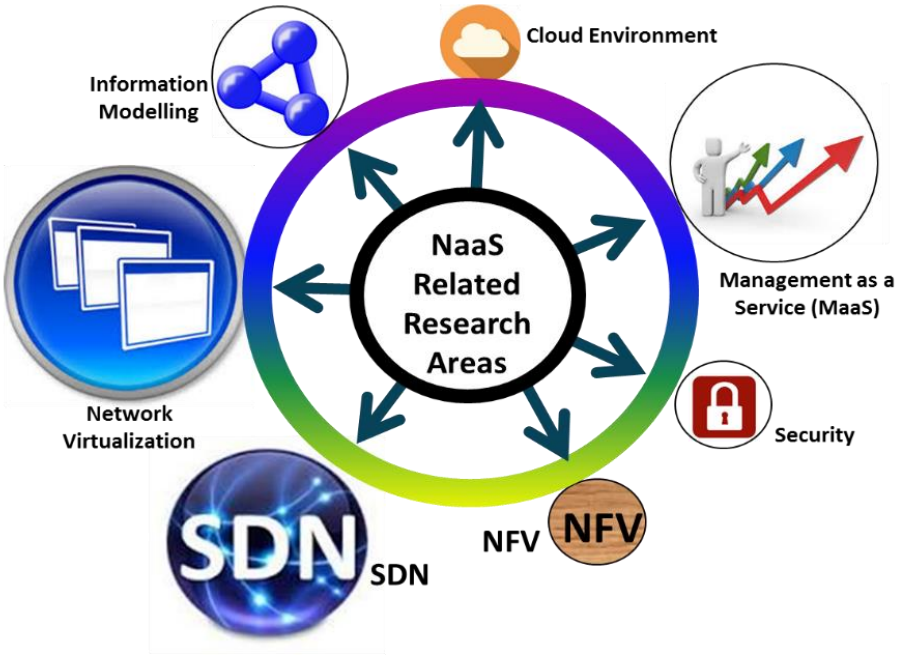

Figure 3.3: Technologies and research areas around NaaS

3.3 NaaS Vision and Roadmap in GÉANT

3.3.1 The Roadmap

The JRA2 Task2 team is focused on proposing NaaS as a service model and developing tangible relevant outcomes in the form of (i) added-value features for GÉANT services and (ii) new PoCs and services of value to NRENs and their community of users. To drive this effort, three main work items were identified:

- Provide a clear definition of the NaaS service model based on the proposed use cases.
- Research specific novel areas of interest with requirements that may be addressed by the NaaS model.
- Define NaaS-based services, starting from identified limitations that NRENs and GÉANT community partners are experiencing while operating their networks. Select and implement PoCs and showcases from previous defined NaaS-based services. Prototype OpenNaaS extensions and demonstrators. Provide a tangible, relevant outcome for the community.

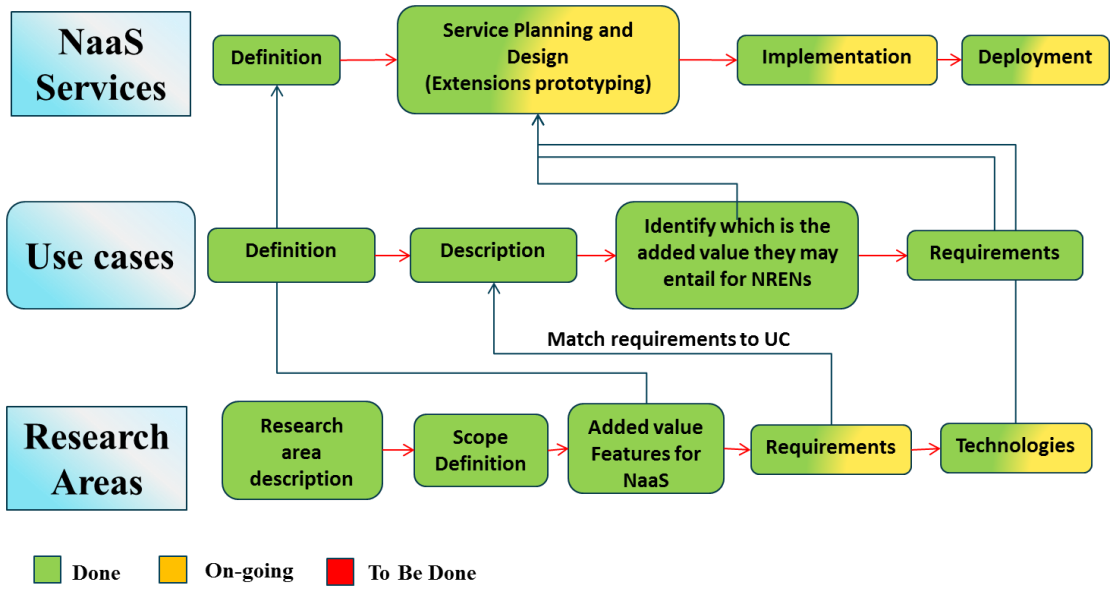

Figure 3.4: RAG rating scheme of most relevant JRA2T2 work items

Figure 3.4 shows a Red Amber Green (RAG) diagram of JRA2T2 in terms of the previous discussed working items being carried out in JRA2T2, and the interdependency and coordination among them. Definition of use cases has been accomplished. Throughout the last quarter of GN3plus, Task 2 partners will continue to research technologies and requirements close to NaaS and extend some of the proposed NaaS services' PoCs.

JRA2T2 has evidenced several limitations while introducing the NaaS model and the PoCs services to the GÉANT community. This has resulted in the need to extend the vision towards a more ambitious one, to reinforce the NaaS message and the services being proposed based on the prototyping of OpenNaaS extensions and demonstrators.

- The examples of use of NaaS so far are all rather limited in time and size of the user base. Demonstrations, proofs of concepts and limited time production setups provide an indication of an upcoming, but still not mature, technology.
- **Emphasis on Management as a Service**: A significant effort has been done analysing state of the art, retrieving requirements for NaaS and integrating both approaches.
- **SDN technology is continuously evolving**: New SDN controllers, frameworks and standardisation efforts are continuously evolving so that the analysis and interworking between SDN and NaaS is ongoing at the moment of writing.
- **OpenNaaS platform flexibility** is not enough to provide NaaS based services. Also, the learning curve and the installation methods have turned out to be quite tricky for new developers and users.

Thus, there are a number of actions foreseen in the NaaS roadmap surrounding the question: "Which actions should be considered to increase GÉANT community involvement and interest in terms of NaaS?" A list of specific items for the last quarter of GN3plus and future projects is provided in the next section.

3.3.2 Foreseen NaaS, SDN and NFV Ecosystem: One Step Beyond

The current ecosystem exposes three main concepts:

- It is possible for architecture designs and systems to be software-defined, enabling one to decouple and separate the data, control, and management and service planes. **SDN technology** enables such features.
- Second, service delivery trends focus on novel features (e.g. rights delegation, more flexible user interfacing). **NaaS service delivery model** embraces this trend.
- Finally, infrastructure and resource management is experiencing forward movements as well, converging heterogeneous infrastructures by means of virtualisation techniques and homogenous device interfacing models. This leans on the **NFV** concept.

The enhancement of current services and capabilities and development of new services (e.g. SDN applications, Management as Services capabilities, NFV as applications, virtualised and fully-programmable network infrastructures) can be achieved by **evolving the NaaS concept and exploiting SDN and NFV capabilities** (see Figure 3.5).

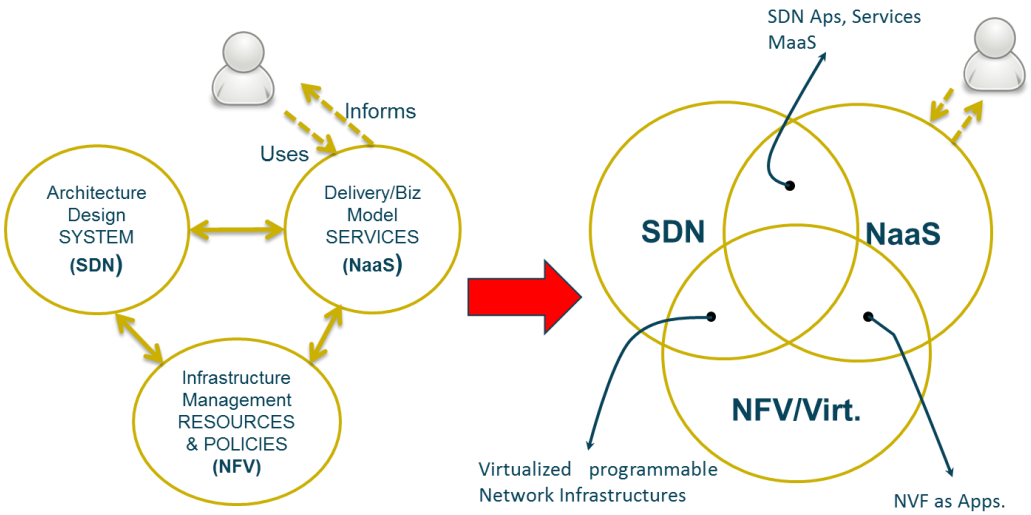

Figure 3.5: NaaS-SDN-VFN current and future ecosystem

3.4 NaaS Research: Key Achievements

3.4.1 Management as a Service (MaaS)

Several new service models emerged from the recent development of cloud and virtualisation technologies. The definition of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a service (SaaS) service models [NIST] is now well understood. These service models were mainly applied to application provisioning IT services over a cloud infrastructure. The main difference between the three models is the placement of the boundary between the service provider and service user responsibilities. Figure 3.6 clearly depicts the difference between the three most common cloud service models.

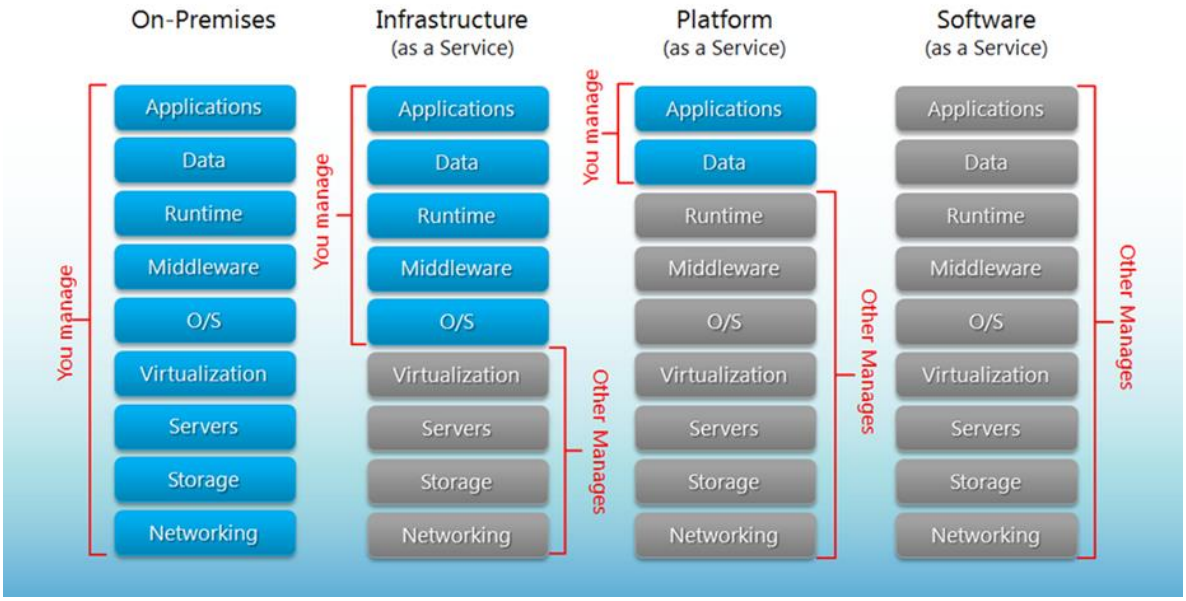

Figure 3.6: IaaS vs PaaS vs SaaS: Separation of Responsibilities

Soon after these service models were developed, the same technological wave, together with some new technologies in the telecommunications sector (the emergence of SDN, I2RS, NFV and similar technologies) influenced the appearance of some new service models, such as Communication as a Service (CaaS), Network as a Service (NaaS) or even Desktop as a Service (DaaS) [ITU-T]. The ITU-T definitions of these terms are provided below as they are not as well known, and various interpretations can be found in the literature:

- Communications as a service (CaaS): A category of cloud services where the cloud service user is provided with use of real-time communication and collaboration services, including voice over IP, instant messaging, and video conferencing, for different user devices.
- Network as a service (NaaS): A category of cloud services where the cloud service user is provided with the use of transport connectivity services and/or inter-cloud network connectivity services. NaaS services include flexible and extended VPN, bandwidth on demand, etc.
- Desktop as a service (DaaS): The cloud service user is provided with the use of virtualised desktops from a cloud service provider in the form of outsourcing.

This section presents a range of views on the Management as a Service in the context of GÉANT and NREN environments as well as from commercial sources, and an accurate definition of the Management as a Service model will be provided.

3.4.1.1 *Management as a Service Definition*

It is not difficult to conclude from the various definitions of "X" as a service (*aaS) service models given at the beginning of this section that a provider provides " * " to the user, where " * " can include different things, such as: infrastructure, e.g. a computer (IaaS), software, e.g. Google Docs applications (SaaS) or network connectivity, e.g. L3VPN connectivity (NaaS). In these cases, the user relies on the provider resources (servers and operating systems, application software or network devices from the previous sentence respectively) and does not own them.

Management as a Service is a category of IT services where the capability provided to the service user is the management of the IT resources and/or services owned by the user. Management functions can be those from the old Fault, Configuration, Accounting, Performance, Security (FCAPS) model, or some more granular processes as those from the TM Forum's enhanced Telecommunications Operations Map (eTOM) model. One clear example of MaaS would be outsourcing the management of IT assets to another company. In the communications arena, the equivalent would be offering some network management functions to a user (e.g. outsourced helpdesk, outsourced network monitoring).

3.4.1.2 *Environment of Management as a Service*

Most present-day technologies virtualise the implementation of a service (such as a data plane being an IP network, an Ethernet connection, a virtual machine, storage, etc.). This service that is actually being produced/used, will be called a Worker Resource (WoR).

Besides the Worker Resource, a Management Resource (MaR) is defined, following the analogy of the management plane in telecommunications. The Management Resource virtualises the operational management functions of the Worker Resource. Adopting the GN3's Network Management Architecture, which is based on the TM Forum's eTOM, the following Management Resources can be recognised:

- Provisioning Management Resource (PrMaR: installing, configuring Worker Resource).
- Performance Management Resource (PeMaR: managing, tracking, monitoring, optimising the performance of a Worker Resource).
- Trouble Management Resource (TrMaR: recognising, isolating and correcting faults).
- Accounting Management Resource (AcMaR: for instance storing information and reporting related to Worker Resources).
- Policy Management Resource (PoMaR: allowing adding, modifying or deleting of policy rules and attributes with regard to the business logic).

The advantage of seeing Management Resources in the same way as Worker Resources is that it opens up the possibility to sell/subcontract, that is, trade, any (Worker and Management) Resources on a market place by using normalised interactions between actors.

As previously stated, MaaS is based on Management Resources and Worker Resources. The relationship between both is many to one, and the proposed design allows for independent Resource manipulation.

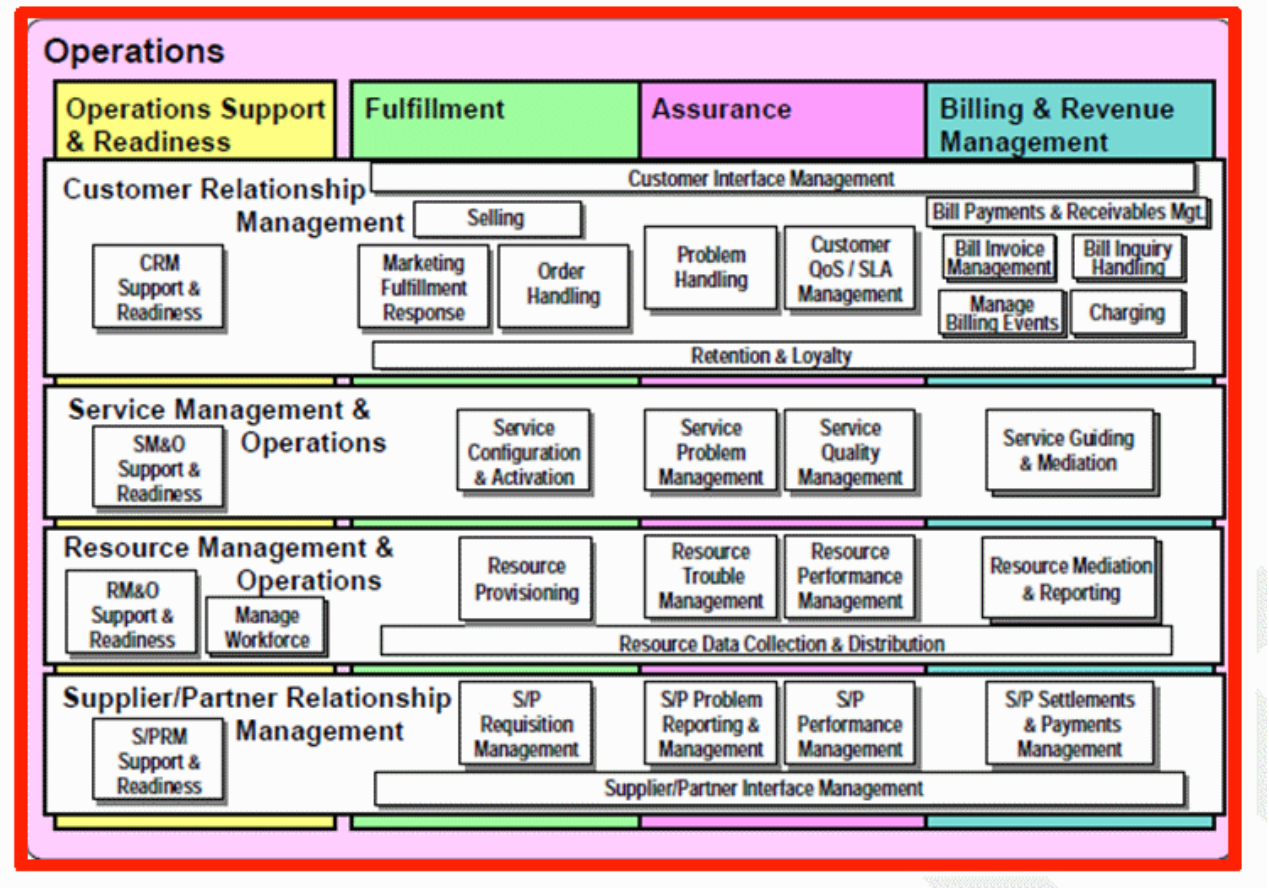

Figure 3.7: eTOM Level 2 business processes in the Operations area

Many vendors offer network management solutions, for instance, ROADM rings (if Web-Services based: Worker Resources) using proprietary, centralised management systems (normally not Web-Services based: Management Resource). Management functions are solidly implemented in a vendor's system. In the MaaS model, the functions are partitioned according to eTOM, and a Management Resource(s) is defined for each. This way, operators' management workflows are more flexible and allowed to incorporate or delegate actions from or to third parties. The management of Worker Resources can also be made more distributed or Management Resources moved into the cloud.

A few use cases can be recognised:

- A researcher is not allowed to manage the security aspect of a Resource: A researcher might have to abide by the security/policy rules of an institute and the institute might not allow delegation of policy to a user, so the policy management has to be done by the IT Service department.
- A consumer does not want to manage anything: The consumer will delegate all the Management Resources to one or more resource operator (depending who is cheaper or better in a certain area).
- A consumer (say an NREN) wants to manage the network itself: The NREN will need access to as many network/IT features possible, so all or most of the Management Resources are managed by the NREN (NREN could still outsource some part of an Management Resources to another resource operator).

3.4.1.3 *MaaS Roles*

The following roles are defined in the MaaS architecture:

- RC: Resource Consumer: the entity that uses the WoR and he/she has the responsibility to allocate the related MaRs to ROs.
- RP: Resource Provider: an entity that provides WoRs.
- RO: Resource Operator: an entity that operationally manages part(s)/whole Resource through the MaRs.

The roles are compatible with those defined in the GEYSERS [GEYSERS] and 4WARD [4WARD] projects.

3.4.1.4 *Management as a Service in the industry*

Although not standardised, the term MaaS is used for some commercial IT services and products. The context in which it is used is not always the same. Here are some examples from the commercial solutions:

- Company CSC offers a product called CSC MaaS – Custom Applications [CSC]. This is application management outsourcing service for custom and COTS enterprise and departmental software applications. This service is offered to support the software products of various software vendors such as Oracle and SAP. The customer requests and receives a requested number of Application Support Units (ASU) which manage applications, work in a 24x7 regime, and is charged upon the number of the ASUs used.
- Fujitsu ITMaaS [Fujitsu] is a set of SaaS-based applications providing the infrastructure, application monitoring and service desk capabilities for delivering efficient IT service operation. It comprises of two main modules: Service desk as a Service and Monitoring as a Service, which are delivered from the Fujitsu Global Cloud Platform.
- GreenPages [GreenPages] offers Cloud Management as a Service (CMaaS) platform for managing assets in the hybrid cloud environment comprising of owned and outsourced resources (on a cloud) tools from multiple vendors, multiple types of clouds (CMaaS). CMaaS Portal allows discovery and asset management, monitoring and alerting, patching, and analysing performance of the resources and services owned by the user, or which user uses from cloud providers, all through the unique interface instead of using various custom management tools.

3.4.1.5 *MaaS in the Research Literature*

Outsourcing network functionalities and features is not a new concept. Enterprises typically want to focus on their core business and are not willing or capable of investing in and maintaining highly skilled expert teams with diverse knowledge (e.g. network protocols, OSs, management methods, network security, servers, etc.) and ever- changing IT equipment to fulfil the needs for connectivity and presence on the Internet. Outsourcing network functionalities often means reduced capital and operational costs. The architecture must also be made more open in order to outsource work. Virtualisation/abstraction/standardising allows for a well-balanced choice and provides also the ability to run virtualised network functions (VNF); such as firewalls, load balancers, routers, switches, etc.) on virtual machines (VM) in the cloud.

Recent advances in virtualisation and cloud technologies, and large bandwidth and small latencies in contemporary networks have inspired some new ideas for offering networks feature-outsourcing services that can elastically grow with the user's demand. Glen Gibb and Justine Sherry proposed two frameworks: Jingling and APLOMB for outsourcing network functions [Gibb et al.] [Sherry et al.]. These architectures assume that in addition to ISPs, there will be both feature providers (FPs) and Resource Providers (RP) that specialise in offering specific network services (e.g. intrusion detection systems, firewalls, proxies, VPN servers or other), which are now typically deployed as middleboxes, and require expert knowledge to be properly maintained, monitored and used. Emerging technologies such as SDN will allow efficient traffic switching towards and from RPs, while new protocols and languages are needed to specify requests and the needs service users could have. Nick Feamster discussed the extension of outsourcing services to home network security [Feamster], while Seyed Kaveh Fayazbakhsh et. al proposed methodologies to assure that outsourcing services are functional as advertised and contracted, have appropriate performance and that the accounting and payment of the services reflect their real use [Fayazbakhsh et al].

Sherry further presents the results of a detailed survey of 57 enterprise networks ranging from small (fewer than 1k hosts) to large (more than 100k hosts). The results of the survey show that the devices that are the most often used in enterprise networks are L3 routers, L2 switches and firewalls. Outsourcing these functions would make the most significant impact to the enterprise network operations and could lead to the biggest savings. Despite the ideas described in Gibb's and Sherry's work, namely, that topological and geographical position of the feature providers and users are not restricted because L3 routing functions are among the core network functions that require high responsiveness and availability [Gkantidis], the most convenient place to outsource these functions is close to the user: in the location of the Internet service provider. Resilience needs to be evaluated depending on the location of the vCPE (virtual Customer Premises Equipment), as putting one's (virtual or physical) CE further away from the client's LAN, might not be workable.

3.4.1.6 *MaaS – Commercial Examples*

Some examples of the commercially available services are provided in this section in order to further explain the MaaS service model and especially to make a distinction between it and SaaS. It is obvious that some companies are offering MaaS solutions which are much closer to the SaaS, and also, the distinction between SaaS and MaaS solutions often lies in the viewpoint of the observer.

Cyan BluePlanet View Case

Cyan BluePlanet View system [BluePlanet] is a new and unique, cloud-based solution for service quality management. The business model of this solution assumes that a number of Measurement Agents from the

BluePlanet or third-party network elements send performance data from user networks to the BluePlanet cloud servers. The data is stored in the cloud, and the Planet View system processes the data and allows the network operator the possibility to view the data, key performance and SLA indicators. The Cyan BluePlanet View system is an example of the SaaS service model where the application that gathers network measurement data and the user interface are hosted and owned by the service provider. The user enables the communication between its measurement agents and measurement collector and uses the user interface to monitor the status of its network connections and services. A user does not have to take care of the measurement repository, or maintain the size of the storage space needed for measurement data, as the service provider takes care of these on behalf of the user.

This system can also be seen as a MaaS solution, where some network management processes are outsourced to the service provider. In this particular example, it can be said that the Cyan BluePlanet View system supports "Resource Data Collection and Distribution" business processes from the eTOM Business process framework [eTOM]. Resource Data Collection and Distribution business process consists of collecting, distributing and processing management information and data, and is only one part of the more general Resource Performance Management business process.

Alternatively, although the Cyan BluePlanet View system is used for Resource Performance Management and Service Quality Management (SQM), and the data it provides is RPM- and SQM-related data, it would be wrong to see it as MaaS for the full RPM or SQM business processes or even for monitoring network performance. The service provider does not offer anything more than Resource Data Collection and Distribution. Although measurement data collection and processing are essential parts of the SQM business process, Cyan BluePlanet View is not an MaaS SQM solution, as in its business model it is assumed that the user is controlling and analysing service and resource performance, managing performance reports, and localising performance problems, which are all key resource and service performance processes. All these processes are not provided by the service provider, but are executed by the user. A MaaS solution for the whole SQM process in this case would assume that service provider is monitoring raw network parameters, correlating them to the SLA parameters for the service that is being managed, detecting performance problems, but also taking corrective action and solving performance problems.

Loggly

Loggly is a cloud-based SaaS platform for processing logs from various devices [Loggly]. It is also an example of an MaaS service model for offering log collection and analysis, which is an essential part of various network management business processes (e.g. problem management). A service user does not have to take care of the storage and software for logging, as that job is offloaded to the service provider.

3.4.1.7 *MaaS Versus NaaS*

As previously discussed, NaaS is defined as a category of services where the capability provided to the service user is to use transport connectivity services and/or inter-cloud network connectivity services. In the NaaS service model, the service provider manages the resources that are used to provide the services (e.g. routers, servers, virtual machines). Those resources are owned (including leased, or obtained through some kind of contract) by the service provider. The service user only gains access to the network path between the requested set of nodes with a required set of service-quality parameters and does not own the network elements on the way between the nodes and does not manage them. It is assumed that the service is managed, because without proper

management the service would not have the required reliability and warranty. However, the provider does not offer to manage any of the users' IT assets.

Thus, a clear difference between the NaaS and MaaS service models emerges. Even more, it seems that there is no function overlap between the two models. The situation, however, becomes a bit more complicated when service-supporting software components are included into this scenario and some ambiguity is possible. Tools such as OpenNaaS are used for rapid and reliable service provisioning and are most likely to be used by NaaS providers (e.g. commercial communication providers, NRENs) to provide a NaaS service to service users (e.g. banks, universities). If such tools are used by the NaaS provider, nothing changes in the basic service model, as everything remains the same for the NaaS service user. What is being changed are the procedures behind the scene, in the provider's offices and data centres. If the service model assumes that the user can use the tool to request the NaaS service, then some management functions are given to the user (e.g. service request handling, service configuration), but still there is no MaaS model, as nothing owned by the user is managed by the provider.

Tools like OpenNaaS can be used in a MaaS service model in the following cases: if the service user owns OpenNaaS, for example, for configuration management of its assets and outsources their management to a provider, or if a provider uses OpenNaaS for the management of the user's resources, then OpenNaaS tool can be seen as a tool for providing MaaS service. These last scenarios will be further explored in the following sections of this report.

3.4.1.8 *Which management functions can be undertaken with the MaaS model?*

In its technical report, Microsoft Research [Microsoft-MaaS] assessed the feasibility of offering network management as a service. Offloading management functions to another entity, which can be off-site, naturally increases latency between the events in the network and responses of the management systems, brings concerns of the availability of the management function, and also brings questions about the security and privacy within the management plane. Therefore, authors define availability and responsiveness of management functions as the key factors that determine whether some management functions are suitable to be moved and offloaded to a management provider (or cloud). Some management functions that require fast, real-time reactions (like routing) are not suitable to be offered using a MaaS service model, while others, such as capacity planning or similar traffic engineering activities that are performed from time to time, do not have any obstacles. Also, some network management functions that require high availability, such as offering authentication or authorisation services have to be carefully designed so that the disruption in connectivity between service user and provider does not affect network users.

3.4.2 NaaS Interworking with SDN

Figure 3.8 shows a generic SDN framework. On the bottom there is the data plane, the task of which is to forward network packets. The data plane can be hardware (switches from well-known vendors with partially disabled control-plane as well as white-label reference architectures of the switching chip manufacturers (merchant silicon) or software-based forwarding, which is often observed in hypervisors. The data plane has no intelligence of its own, so it needs specific instructions on how to handle packets received by the system. This information is encoded in flow tables, which define the handling of packets by the use of filters (matches) and actions. These flow tables are provided by the controllers via the **southbound interface**. This interface often uses the OpenFlow specification (or alternatives) as an industry-wide standard for the exchange of flow tables, statistics and topology

information. The data-plane is usually not aware of tenants; communication is entirely controlled by the setup of the flow tables.

The controller forms the “common intelligence” of the SDN. It is aware (by virtue of discovery and configuration) of topology, bandwidth and capabilities of the data-plane as well as the users connected to the data-plane and their requirements. The controllers can perform the control plane functions such as switches, routers, load balancers or firewalls on their own, as well as provide entirely new functionalities that do not match the classic interaction of ISO/OSI-layers. The controllers are aware of virtualisation as it appears in the context of NaaS and program the separation into the data plane.

Because of redundancy and mostly because of the total amount of switches, there may exist multiple controllers inside a single network. These controllers have to communicate with each other, e.g., to provide failover capabilities or to synchronise flow rules. This inter-controller communication is done via the **east-west interface**.

The controllers can be influenced via the **northbound interface**, which should provide a standardised way for external applications to request services or information from the controller. This allows innovative applications to be developed on top of a SDN-based service provider, which does not necessarily have to understand the application in either the controller or in the data plane. One example in the NaaS context could be a broker service that is capable of setting up a multi-provider VPN service by talking to the controllers of all service providers in the path. This proliferation of SDN controllers, each with unique APIs capabilities at the northbound interface creates an overabundance of programmatic interfaces that network service providers, orchestration systems, application vendors, and software developers at large must implement in order to support diverse SDN use cases. This is slowing down the widespread adoption of SDN in both data centre and transport network ecosystems, as well as the usage of its associated protocols and technologies, including, but not limited to, OpenFlow [OpenFlow].

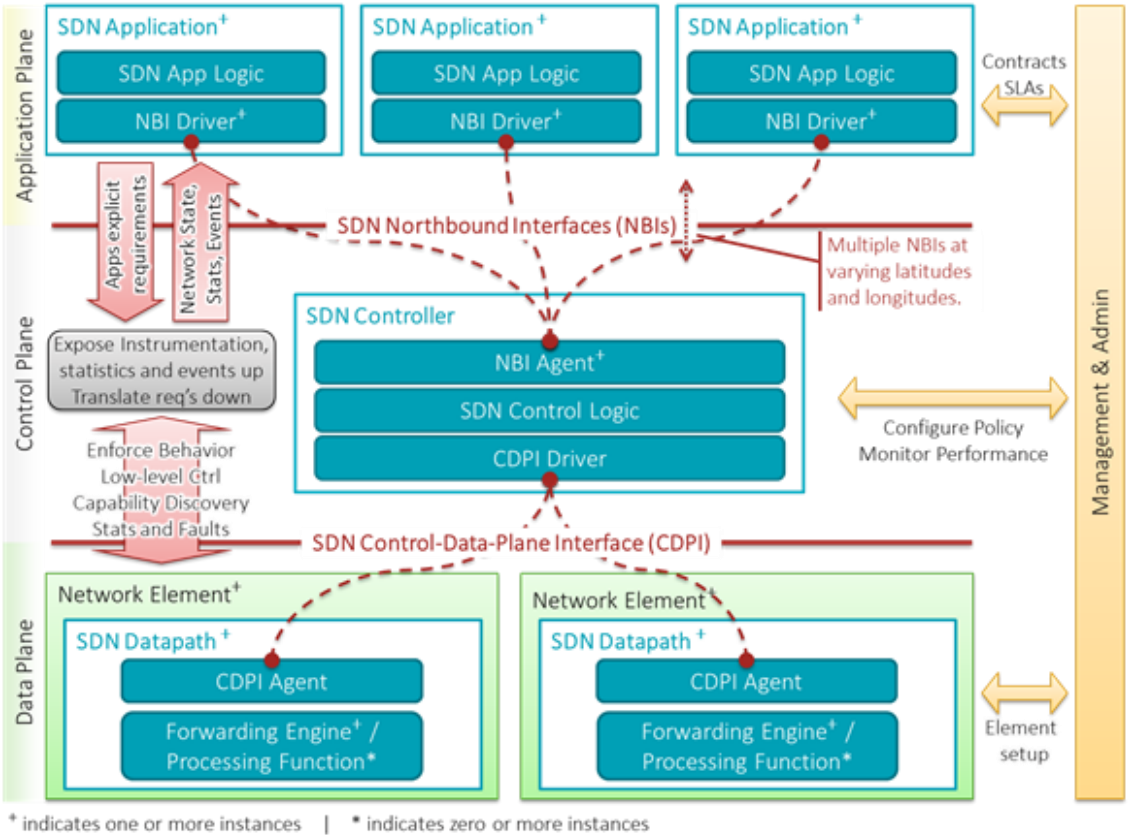

Figure 3.8: SDN Framework Overview. Source file retrieved from [SDN-Arch]

GN3plus JRA2 Task 1 is conducting a full analysis of the current SDN panorama and the impact that this will have on GÉANT, NRENs and their user communities. Thus, the scope of SDN in the context of JRA2T2 is not to overlap with such analysis, but to research the potential interworking and coordination between NaaS (as service model framework) and SDN (as underlying network technology).

The research effort has a twofold objective:

- Analysis of the SDN northbound interface (NBI) definition status and the standardisation actions carried out to the date. The SDN northbound interface is the point of connection to coordinate both SDN and NaaS layers. Most of the relevant standardisation approaches have been taken into account and a preliminary list of NBI requirements has been compiled. Multiple vendors are bringing commercial SDN products to market, and a lot of SDN controllers are currently available, each featuring different northbound capabilities and mechanisms. The main reason for such variation is that the requirements for the northbound APIs vary, depending on the needs of the specific applications and orchestration systems on top of each SDN controller solution, which, in turn, are often strictly related to the hardware and technology choices at the network level. As a consequence, a reference standard for SDN northbound APIs is defined in [LD43].
- Analysis of the SDN SBI (Southbound Interface). This work is being addressed based on a cross-collaboration between T1 and T2. The main point is to determine "how the NaaS layer may leverage/consume the information provided by the SDN layer". The analysis has been conducted for the following SDN controllers: FloodLight, RYU and OpenDaylight. During the remaining GN3plus months, partners aim to extend the scope also to ONOS controller.

A thorough analysis of most representative SDN controllers has been performed. The most relevant conclusions on NBI and SBI are provided here.

3.4.2.1 *Standardisation Attempts*

Figure 3.9 shows some challenges in the standardisation of the northbound interface. This is only a simple scenario, where both of the execution contexts are the same. The issues identified are only indicative and raise the question on whether there can be one standardised northbound interface API or if there should be multiple, e.g., scenario based reference models. The most representative standardisation efforts on northbound interfaces are presented in the next sections.

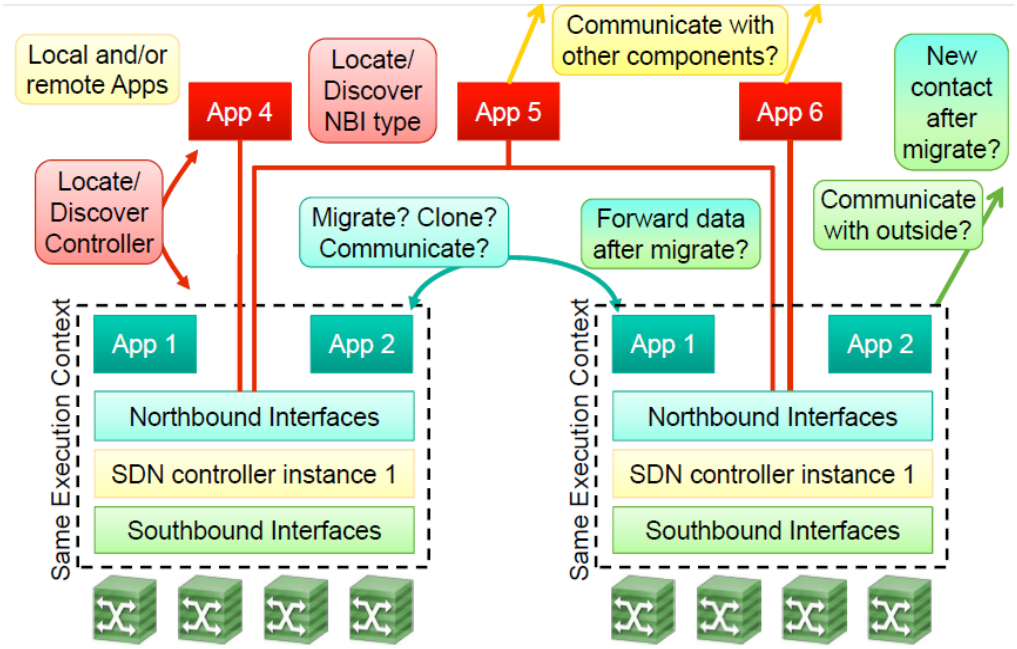

Figure 3.9: Standardisation challenges for SDN northbound interface [SDN-NBI-Std]

ONF NBI WG (ONF NorthBound Interface Working Group)

The Open Network Foundation (ONF) started the ONF SDN NorthBound Interface Working Group in November 2013, which has the goal to help develop concrete requirements, architecture, and working code for northbound interfaces [ONF-NBI-Charter]. The ONF aims to reduce end-user confusion on the northbound API and define an open API for application developers. The main goals of the ONF NBI WG can be therefore summarised as follows [LD43]:

- Provide extensible, stable, and portable northbound APIs to controllers, network services, network providers and operators, application developers.
- Increase the portability of those network applications designed and implemented to interact with SDN controllers. To this end, multiple APIs need to be enabled and defined at differing levels of abstraction in support of enhanced programmability and automation for network configuration, provisioning and monitoring.
- Encapsulate the complexity and details of the underlying southbound implementations to make the northbound APIs available to a larger number of applications.

- Ensure that the defined and standardised northbound APIs allow SDN controller vendors to freely innovate, using specific and customized API extensions, their products and solutions.

The ONF SDN NBI Community SDN Northbound Interfaces Community Site [ONF-SUB] has been open since August 2014 to collect functional requirements, but so far, no documents have been posted.

IETF: Forwarding and Control Element Separation (ForCES), framework analysis, main characteristics, conclusions.

The Request for Comments 3746, proposed the architectural framework for the ForCES (Forwarding and Control Element Separation) network elements, and identified the associated entities and their interactions [RFC 3746]. In this framework, which predated SDN, the forwarding and the control elements have to be separated. There is also communication between the controller element and some network application through the Agent Extensibility (AgentX) Protocol, defined in RFC 2741, and mainly based on SNMP queries. Even if ForCES is also not fully standardised, the protocol in use is defined [RFC 2741]. There have been no drafts published in the ForCES working group that discuss the northbound interface towards applications.

3.4.2.2 *SDN Controllers NBI and SBI Interfaces Survey*

Among the large variety of SDN controllers, a set of the most representative have been selected for analysis in order to understand the options that NBI and SBI interfaces provide to NaaS and make a list of the minimum requirements that would be desirable.

3.4.2.3 *SDN Controllers Survey*

Floodlight

At the northbound API, a set of applications is available using a REST-based protocol. These include:

- Virtual Switch, which is an application for creation of Layer 2 virtual networks in a single Layer 2 domain.
- Circuit Pusher allows the user to create bidirectional circuits by updating flow tables on the target devices.
- ACL (Stateless FW) is an application that enables the enforcement of access control lists for the in-packets. It supports simple actions such as allow/deny.
- OpenStack Quantum exposes NaaS model through REST API that Floodlight implements.

At the southbound interface, Floodlight currently only supports the OpenFlow 1.0 specification. The timeline for support of newer versions (such as OpenFlow 1.2/1.3) is unknown. With only OpenFlow 1.0 supported, the information that can be retrieved from the data plane is severely limited. It supports the retrieval of port and flow throughput, as well as basic inventory and topology information. A number of third-party projects built on top of Floodlight are available, which implement additional features. Nevertheless, the development of Floodlight seems to have stalled, with the latest release (0.90) shipped in September 2012, and the latest commits in the GitHub repository around June 2014. The last period of higher activity on GitHub has been in July 2013.

FloodLight Utility for NaaS-Related Purposes

Floodlight is an easy to use controller, however, the applications shipped with Floodlight do not support any multi-tenancy and are thus unusable in a NaaS context. Floodlight was included in this comparison based on its popularity among SDN beginners.

Conclusions

Floodlight has been a reference implementation in the early days of SDN development. It still provides a very good implementation of basic SDN-features, but the sponsor (Big Switch Networks) seems to be concentrating on its commercial offering of SDN controllers for specialised applications (like the BigTap Monitoring Fabric solution) and the commercial variant of the Indigo Data Plane called Switch Light, which implements OpenFlow 1.3 on both Broadcom-based bare-metal switches and hypervisor-based OVS installations.

Ryu

Ryu is a component-based SDN framework fully written in Python. It comprises components accessed by a defined API. These include basic components for an SDN controller, as well as components that are useful for SDN applications on top of the controller. The architecture includes a southbound SDN and legacy interfaces, as well as northbound interfaces based on WSGI.

Ryu offers a **northbound** RESTful API with JSON data format for access to information and configuration. A subset of the supported APIs can be found in the official Ryu-book [RYUBook]. Generally, every module can offer its own REST API, with the API documentation often found in-line with the source-code. The shipped applications can be found at GitHub [GitHub] and include:

- Retrieving information for the controller, connected switches or single switchport, as well as adding/modyfing/deleting of flow entries (ofctl_rest.py).
- Topology information (rest_topology.py).
- Endpoint Information (rest.py).
- QoS settings (rest_qos.py).
- Addresses and routing tables of the router module (rest_router.py).
- Retrieving status and logs as well as adding and removing firewall rules (rest_firewall.py) northbound interface analysis.

Regarding the **southbound** interface, Ryu supports various protocols for managing network devices, such as OpenFlow, OF-CONFIG and NETCONF. For OpenFlow, Ryu fully supports 1.0, 1.2, 1.3, 1.4 and Nicira Extensions. Supported non-SDN protocols include NETCONF, VRRP, xFlow and the Open vSwitch Database Management Protocol (OVSDB). Internally, Ryu provides a packet parsing library supporting a wide variety of TCP/IP stack protocols which can be accessed by the SDN apps and leverage many existing legacy data sources, e.g. OSPF, BGP, MPLS and LLDP. Other protocols such as STP and LACP are also supported. In terms of hardware switch interoperability, Ryu is referenced by some switch vendors, and it supports Open vSwitch (OF1.0, OF1.2, Nicira extensions, OVSDB).

Ryu allows the retrieval of the following characteristics from the data plane (e.g. in the SDN-Starter-Kit: attached network components and their ports, flow and port statistics (received/transmitted bytes/packets, drops, errors).

Ryu use for NaaS-Related Purposes

Ryu's out-of-the-box components allow for a wide range of SDN and non-SDN legacy network configuration/status information sources. This promises a fully SDN-driven management of the network.

Conclusions

RYU is quite flexible and versatile, and despite its many components, new applications can quite easily be developed in python or alternatively, new components can be created. Ryu has support for all important legacy data protocol sources (in case of SNMP at least e.g. via other Python libraries) and supports many further OpenFlow-related protocols, e.g. OF-CONFIG. Furthermore, it provides, out-of-the-box, a wide range of applications, e.g. RESTful router or firewall application, which can at least serve as examples for new developments. It integrates with Snort, Zookeeper, and OpenStack Quantum (now renamed Neutron). It should be noted that Neutron and Ryu also have a plugin, which features L2 isolation and multi-tenancy without or with VLAN, which may be a promising component for NaaS realisation. Concerning e.g. QoS, reliability can be increased by use of built-in supported LACP. Above all Ryu comes with extensive testing support (unit testing, packet generation). In summary, Ryu provides all of the most important information required in for NaaS-SDN and can serve as the basic platform.

OpenDaylight

OpenDaylight is a Linux Foundation project that has very successfully acquired sponsorship by major network component, hardware, and software vendors, including Cisco, HP, IBM, VMware, and Microsoft.

The OpenDaylight **northbound** interface offers a comprehensive list of REST APIs provided in its Wiki [ODLWI]. The API is grouped into several functionally separate modules that include:

- Statistics REST API for retrieving flow-, port- and table-statistics.
- Topology REST API for retrieving topology information as well as adding/deleting user configured links.
- Static Routing REST API for adding, modifying and deleting static IP routes.
- Flow Programmer REST API which permits the configuration of static flow entries.
- Bridge Domain REST API for configuration of the included L2 switch.
- Host Tracker REST API allows querying for the location of an IP address in the network, which essentially represents the switch and port where the host is connects.
- The API also includes a whole class of OpenStack Neutron Network Configuration APIs for the OpenStack-like NaaS topologies.

An interesting feature of the OpenDaylight API is the concurrent support for both JSON and XML-formatted data for both input and output. The JSON format allows rapid prototyping and is easy to read, while the XML format can be validated using the shipped XML-Schema. Unfortunately, most fields are defined to be strings in the Schema, which makes it less useful for validation purposes.

Taking a look on the **southbound** interface, OpenDaylight can be used with a broad spectrum of network devices, including OpenFlow-enabled systems, Open vSwitch, and traditional network components that are managed via SNMP. Additionally, it allows for the integration of BGP-based routing information and includes the aggregation and refinement of statistical data as a part of its base network service functions, which, in turn, allows for a data-plan-independent forwarding of relevant information to the upper NaaS layer.

As OpenDaylight is Open Source software (Eclipse Public License, EPL) and has a modular, plug-in-supporting architecture, any information retrievable through management protocols, such as OpenFlow or SNMP, can be obtained from the underlying data plane resources, especially the SNMP4SDN package. SNMP4SDN is a part of OpenDaylight, and is a powerful candidate for migration towards SDN as it allows to apply SDN-based management principles to off-the-shelf Ethernet components, including configuration (e.g., VLANs, STP, flooding protection), interface statistics, and topology information (e.g., via LLDP).

In general, OpenDaylight allows retrieval of the following characteristics from the data plane: Attached network components and their ports, transmission rates, flows, static routes, tracked hosts, subnets, flow and port statistics (received/transmitted bytes, packets, drops, various error types, collisions).

OpenDaylight Suitability for NaaS-Related Purposes

The main advantage of OpenDaylight in the context of NaaS–SDN is the amount of information that can be made available from the managed network components and the option to integrate legacy components by making use of the SDN-to-SNMP gateway. This allows a fully SDN-driven management of the network without sacrificing the option to use legacy management tools, e.g., for performance management and SLA fulfilment monitoring, and can serve as a basis for NaaS-/SDN-specific routing algorithms.

Conclusions

OpenDaylight is a highly interesting, but also complex platform that comes with a steep learning curve. It currently is being more actively developed than Ryu, but is not as focused on the pure SDN controller as Floodlight. OpenDaylight can provide all the information required in the NaaS-SDN context, but still needs testing for stability outside small lab environments.

ONOS

The Open Network Operating System (ONOS) is a new SDN controller developed by ON.LAB. The development is funded by service providers (AT&T, NTT Communications), network vendors (Ciena, Ericsson, Fujitsu, Huawei, Intel, NEC), R&E network operators and other stakeholders. ONOS claims to be one of the first SDN controllers designed to be a foundation for commercial products and “Industry’s First Open Source SDN Network Operating System for Service Providers”. In order to achieve that the architecture defines the following key features [ONOS-WP]:

- Northbound Abstraction/APIs that include network graph and application intents to ease development of control, management, and configuration services.
- Southbound Abstraction/APIs that enable pluggable southbound protocols for controlling both OpenFlow and Legacy devices. The southbound abstraction insulates the core of ONOS from the details of different services and protocols.

Utility for NaaS-Related Purposes

Source code and extensive documentation were released on 5 December 2014, which makes it impossible to gauge the architecture and the API coverage comparable to the other contestants.

Conclusion

ONOS claims to be nothing less than the holy grail of SDN controllers. It is impossible to confirm or reject that claim before the code has been extensively tested. According to ONOS' whitepaper, the architecture addresses key requirements for SDN deployment in carrier-grade production networks, such as scale-out and redundancy. ONOS is a promising offer that still needs to prove the claims in real-life tests.

3.4.2.4 *Northbound Interface Requirements*

Despite the OpenFlow-based standardisation of the southbound interface and the additional possibilities offered by frameworks such as OpenDaylight, e.g., gateways for traditional SNMP-based management, the obstacle remains that the northbound interfaces are still vendor- and implementation-specific.

While it is a good starting point that Open Source software such as Floodlight already integrates NaaS capabilities, the lack of a standard still endangers the long-term NaaS operation because GÉANT management tools will need to rely on stable interfaces. Even though Floodlight enables several interaction options, such as directly using the Floodlight applications like Circuit Pusher, or accessing the Floodlight core API. These options are expected to change over time.

Therefore, a more generic and stable solution is the integration of a novel medium layer located between the NaaS layer and the SDN controller. This new layer, referred to as the Common SDN Adapter, serves as a stable interface to the NaaS layer and implements the drivers for accessing various SDN controller implementations in a modular way. This architecture could serve as a basis, e.g., to connect OpenNaaS to both Floodlight and OpenDaylight.

The following data is representative of the minimum set that needs to be provided to the NaaS layer through this adapter layer:

- Lists of ports, flows, aggregations, tables, inter-component links.
- Component description.
- Traffic and error counters.
- Tracked devices including MACs, IPs and ports.
- Queue parameters: minimum and maximum rates.
- Port settings: no-receive, no-forward, admin-state, speed, duplex mode, medium, auto negotiation, pause.
- Tunnel endpoint data (e.g., GRE, VxLAN, NVGRE).

3.4.2.5 *Enhancing NaaS Based on SDN*

This section describes the major enhancements that can be achieved by using SDN for NaaS:

- **Programmable network appliances (both physical and virtual)**: SDN switches can be programmed to not only offer packet-forwarding behaviour, but also work as firewalls or load balancers. Such instructions can be realised with just a few lines of code. This programmability also allows joining or splitting networks in an ad-hoc manner, which provides flexibility from both the customer and the

administrator perspective. It also means that network appliances could be replaced by unified programmable and multi-purposed SDN devices, which simplifies the management and maintenance while lowering operational expenses at the same time.

- **Refined NaaS control of network resources**: Management rules and policies can be explicitly stated and enforced using proper configuration or programming languages, such as Maple or Frenetic. This can dramatically decrease the overhead for administrators when configuring complex user requirements. Policies could even be negotiated interactively with the users to adapt them to their specific needs.
- **Advanced analysis of network behaviour**: SDN supports in-network information gathering and analysis in order to gain deeper insight to network behaviour and events in real-time. For example, counter values such as packet-in/-out can be sent from the device to the controller without performing interval-based polling. This enables on-demand pushing of information from the southern layers to the NaaS layer.
- **Incorporating QoS into on-demand networks**: QoS requirements can be mapped to rules in flow tables so that QoS policies are reflected and realised, for example, the control of bandwidth for NaaS virtual networks with different quality and load-balancing requirements could be enforced using SDN rules expressed in the form of functions, as in conventional programming languages.
- **Continuum in network management**: Continuum here refers to the implementation and refinement of management policies or rules from high-level perspective to low-level executable management functions. This is difficult to achieve using traditional network technology due to a lack of unified methods. SDN policy languages, such as Pyretic, allow the generic expression of network management intentions in terms of a unified programming language and supporting scalability by providing structuring concepts such as classes and functions.
- **Intelligent network management with SDN**: SDN is a flexible, intermediate layer between the high-level services offered by NaaS and the low-level infrastructure components.

3.4.3 Information Modelling

Information modelling aims to improve OpenNaaS using semantic Web technologies. The main advantage of modelling OpenNaaS networks in ontologies is that it can model various network components in a vendor-independent form. That leads to adding a semantic dimension to OpenNaaS, and subsequently, to a Network as a Service paradigm. That includes several potential improvement aspects for OpenNaaS:

- Support of network request validation and network connectivity check.
- Capability of constructing distributed and interconnected OpenNaaS instances.
- Complex status reports generation.

At the core of the information modelling is the Network Mark-up Language (NML) ontology. First, the NML ontology is described in detail, and then a description of the improvements Semantic Web adds to OpenNaaS follows.

3.4.3.1 *Ontology*

The ontology is at the core of information modelling, as it constitutes the main concepts required to create, define and link the various network nodes. The Network Markup Language (NML) ontology [NML-Schema], [NML] has

been reused and improved. More classes and properties have been devised, and the existing ones enhanced, in order to permit the creation of complex network topologies.

Table 3.2 and Table 3.3 describe the most significant ontology classes and properties, and describes how to use them for creating topologies in OpenNaaS.

Concept	Parent	Purpose
nml:NetworkObject	owl:thing	It is the parent class of most classes of the NML ontology. It denotes a generic network object.
nml:Node	nml:NetworkObject	A Node is a device connected to a network. It can refer to a physical or a virtual device.
nml:Port	nml:NetworkObject	A Port constitutes the connectivity interface of a NetworkObject to the rest of the network. The port is unidirectional, and it does not necessarily refer to a physical interface.
nml:Link	nml:NetworkObject	A Link object describes a unidirectional data transfer from each of its sources to all of its destinations (sinks). The source of a Link is NetworkObject and its sink is another NetworkObject.
nml:LinkGroup	nml:Group	LinkGroup is simply a collection of unordered links.
nml:Topology	nml:Group	A Topology is collection of connected NetworkObjects, i.e. there is a connection path between any two NetworkObjects belonging to that topology.
nml:Lifetime	owl:thing	The Lifetime represents an interval during which a NetworkObject is active.

Table 3.2: Most important concepts of the NML ontology

Property	Domain	Range	Purpose
nml:isSource	*nml:NetworkObject*	Link LinkGroup	*isSource* relates a *NetworkObject* to one Link to define its incoming traffic *NetworkObject*.
nml:isSink	*nml:NetworkObject*	Link LinkGroup	*isSink* relates a *NetworkObject* to a *Link* to define its outgoing traffic *NetworkObject*.
nml:hasInboundPort	*nml:NetworkObject*	nml:Port	Relates a *NetworkObject* to its respective *Port* declaring that the direction of flow is towards that *NetworkObject*.
nml:hasOutboundPort	*nml:NetworkObject*	nml:Port	Relates a *NetworkObject* to its respective *Port* declaring that the direction of flow is away from that *NetworkObject*.
nml:start	*nml:Lifetime*	xsd:dateTime	The start time of a *Lifetime*.
nml:end	*nml:Lifetime*	xsd:dateTime	The end time of a *Lifetime*.

Table 3.3: Most important properties, domain and range of the NML ontology

3.4.3.2 *New Features available by Semantic Web*

Request Validation and Connectivity Checking Component

A request needs to be formulated to action a demand for the instantiation of a network topology. This request contains information about the various network components, e.g. ports and links, that should be created by OpenNaaS, as well as how those components are interconnected. The NML ontology is used to properly formulate a request. In addition to providing taxonomy of the various network objects, this ontology plays the role of the controlled vocabulary used to create a request, along with its inherent components, e.g. links.

Two levels of validation may be performed: a request validation, and a connectivity check. The first level ensures that the request is syntactically correct, i.e. no syntactic error exists in the request. Further, we use Pellet reasoning [Pellet], to validate the request against the NML ontology. In other words, it checks that the request adheres to the NML vocabulary, and no violation exists, e.g. no component is declared as a port and as link at the same time, as those concepts are disjoint.

The second level of validation is the connectivity validation, in which we validate the request and detects if a network node or a port is unreachable from another node or port. That helps in detecting the inaccessibility of some components at that early stage, which helps in avoiding that problem when the resources are allocated. We use SPARQL [SPARQL] queries to detect any unreachability problems among the various nodes of the requested network.

After the request is validated, it is passed to OpenNaaS for provisioning. Moreover, the request is saved in RDF to Virtuoso triple-store [Virtuoso], for later use. That is very useful for report generation, and for system utilisation analysis, i.e. it helps in tracking how OpenNaaS is used, and estimating how long it takes till the resources are provisioned to the user.

Interconnection of Distributed NaaS Instances

Semantic Web supports the establishment of multiple distributed NaaS instances and interconnection of their respective components. One of the major challenges that OpenNaaS has faced is how to maintain the uniqueness of the IDs assigned to the network components in order to interconnect them whenever required. In other words, each network component, e.g. network node or port, has an ID that is only unique across the boundaries of the OpenNaaS instance in which it resides. This introduces an obstacle in connecting the components of two or more remote OpenNaaS instances.

Semantic Web depends on using URIs (Uniform Resource Identifier) as an enabling technology. Since the NML ontology is used to describe networks provisioned by OpenNaaS, we should assign a unique URI to each component. This URI plays the role of its global identifier across all OpenNaaS instances.

In GÉANT's case, *http://ivi.fnwi.uva.nl/sne/* and *http: //www.i2cat.net/* have been used as base namespaces for the OpenNaaS instances hosted in the University of Amsterdam and i2CAT, respectively. For example, the respective URIs for node1 will be *http://ivi.fnwi.uva.nl/sne/resource/node1*, and *http://www.i2cat.net/resource/node1*. Thus, those nodes instantiated by distant OpenNaaS instances can be interconnected without any problem.

Report Generation Component

The aforementioned information model empowers OpenNaaS to create complex reports about the whole resource reservation process. Those reports enable the system administrator to identify the problematic resources of OpenNaaS, monitor the resource failure conditions, and identify the lifetime of any resource, i.e. the time from its creation until release.

The nodes are connected to a Lifetime, to indicate the duration of reservation of resources, which are assigned to a specific user. Based on that duration values, the OpenNaaS administrator can generate complex reports showing the reservation time of some network resources, when each reservation has been established and assigned to the user, whether it is still active, etc.

Conclusions

Using ontologies to describe network resources managed by OpenNaaS is beneficial. It has several advantages and capabilities that can add to OpenNaaS:

- The ability to validate a request and check the reachability of the various components.
- Complex report generation, including the lifetime of each resource.
- The capability to connect several remote OpenNaaS instances, as each resource will be assigned a unique Uniform Resource Identifier (URI).

3.4.4 Virtualisation Techniques and NFV

3.4.4.1 *NFV Definition and Identified Features*

NFV is the virtualisation of network services. NFV came to fruition when service providers attempted to speed up deployment of new network services in order to foster revenue and growth plans. Noticing that hardware-based appliances limited their ability to achieve these goals, they looked to standard IT virtualisation technologies and discovered that NFV helped accelerate service innovation and provisioning. This led to the creation of the European Telecommunications Standards Institute (ETSI) NFV ISG, which set the NFV basic requirements and architecture.

From an innovative/technical point of view, NFV has introduced [Jain13]:

- **Software implementation of network functions**, i.e. virtualisation: The possibility to create network virtualised resources and allocate them where needed.
- **Network function modules**: Such modularity enables a better performance and distribution of network services.
- **Standard APIs between modules** which make possible a more flexible integration control and service composition based on NFV chaining procedures.
- **Orchestration**: The virtualised approach enables management of thousands of devices and orchestration of them to compose more complex network services.
- **Network Programmability**: Potential to handle CRUD procedures with network functions enables the management of fully programmable environments.
- **Dynamic scaling**: Related to network programmability, the number, sizes and quantity of NFVs is more dynamic than while using of HW facilities.
- **Automation**: Network service automation based on NFV chaining is one of the more valuable features.
- **Openness**: Full choice of modular plugins (OPNFV).

From a business perspective, the move to NFV also increases cost efficiency and agility and opens up new opportunities. The most relevant identified ones are:

- Capex and opex reduction by automating processes such as application deployment, maintenance and capacity planning.
- Faster time to market and faster ramp-up of new services and functionalities.
- Agility and flexibility while delivering network functionality.

This brief introduction to NFV is just a sample of the relevance that NFVs bring to the scientific community and the industry. NFVs development, deployment and management are highly aligned to the NaaS service delivery model and, together with SDN, may define the wave of new IP-based infrastructures.

3.4.4.2 *NFV Standardisation Attempts and Current Panorama*

ETSI NFV ISG – ETSI NFV Industry Specification Group (ISG)

The ETSI NFV ISG was established by charter, not as a SDO (Standards Development Organisation), but as a body that would specify a broad set of requirements for an NFV platform that can be adopted by diverse network operators whose environments span a wide range of scale, operational, regulatory, and technology requirements [SDNCentral]. The ISG defined Virtual Network Functions (VNFs) to refer to the virtualised network software functionality. The TSC (Technical Steering Committee) is split into four WGs (Working Groups) and two EGs (Expert Groups) as shown in Figure 3.10.

- INF WG: Architecture for the virtualisation Infrastructure.
- MANO WG: Management and orchestration.
- SWA WG: Software architecture.
- REL WG: Reliability and availability, resilience and fault tolerance.
- Security Expert Group: Security.
- Performance and Portability Expert Group: Scalability, efficiency and performance of NFVs relative to current dedicated hardware.

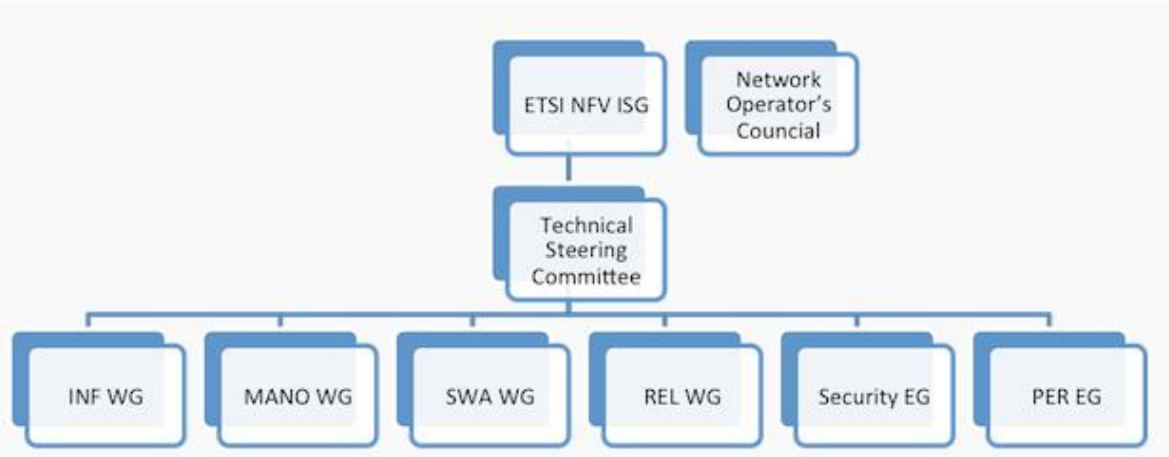

Figure 3.10: ETSI ISG Operational structure in accordance with ETSI conventions [SDNCentral]

The ETSI NFV ISG is currently focused on several end-to-end documents that form the baseline for subsequent technical working group activities:

- NFV Use Cases: A representative set of service models and high-level use cases that may be addressed by NFV [GSNFV001].
- End-to-End NFV Architecture: A high-level functional NFV framework and reference architecture that ultimately serves to partition out the NFV work activities [GSNFV002].
- VNF terminology for Main Concepts in NFV [GSNFV003].
- NFV Virtualisation Requirements: High-level operational requirements for to achieve the goals of NFV, broken down into the NFV technical domains [GSNFV004].

- NFV Proof of Concepts Framework: it defines a framework for use within ETSI NFV ISG to coordinate and promote public demonstrations of Proofs of Concept (PoC), illustrating key aspects of NFV [GSNFVPER002].

Linux Foundation – OPNFV

In September 2014, the Linux Foundation announced another open source reference platform: the Open Platform for NFV Project (OPNFV). OPNFV is a new open source project focused on accelerating the evolution of Network Functions Virtualisation (NFV). OPNFV will establish a carrier-grade, integrated, open source reference platform that industry peers will build together to advance the evolution of NFV and to ensure consistency, performance and interoperability among multiple open source components. Because multiple open source NFV building blocks already exist, OPNFV will work with upstream projects to coordinate continuous integration and testing while filling development gaps.

The initial scope of OPNFV is to build an NFV Infrastructure (NFVI), Virtualised Infrastructure Management (VIM), and including application programmable interfaces (APIs) to other NFV elements, which together form the basic infrastructure required for Virtualised Network Functions (VNF) and Management and Network Orchestration (MANO) components. OPNFV is expected to increase performance and power efficiency; improve reliability, availability, and serviceability; and deliver comprehensive platform instrumentation.

OPNFV will work closely with ETSI's NFV ISG to drive consistent implementation of standards for an open NFV reference platform. Increasingly, standards are being drafted in conjunction with major open source projects [OPNFV].

Open Networking Foundation (ONF)

The ONF is also supporting the NFV, especially in the context of OpenFlow enabled SDN networks and NFV. In [ONF-NFV] an overview on SDN technology and the fitting of ETSI NFV principles to provide a flexible solution and a set of NFV requirements close to SDN ones is provided.

3.4.4.3 *NFV Reference Architecture Model and Relation to the NaaS*

The NFV framework (Figure 3.11) proposed by ETSI is composed of three main components [GSNFVPER002]:

- **Virtualised Network Functions (VNF):** software implementations of network functions that can be deployed on a Network Function Virtualisation Infrastructure (NFVI).
- **NFV Infrastructure (NFVI)**: is the totality of all hardware and software components that build the environment in which the VNFs are deployed. The NFV-Infrastructure can span across locations. The network providing connectivity between these locations is regarded as part of the NFV-Infrastructure.
- **Network Functions Virtualisation Management and Orchestration Framework (NFV-MANO Architectural Framework): MANO** is the collection of all functional blocks, data repositories used by these functional blocks, and reference points and interfaces through which these functional blocks exchange information for the purpose of managing and orchestrating NFVI and VNFs.

The building block for both the NFVI and the NFV-MANO is the NFV platform. In the NFVI role, it consists of both virtual and physical compute and storage resources, and virtualisation software. In its NFV-MANO role it consists

of VNF and NFVI managers and virtualisation software operating on a hardware controller. The NFV platform implements carrier-grade features used to manage and monitor the platform components, recover from failures and provide effective security – all required for the public carrier network.

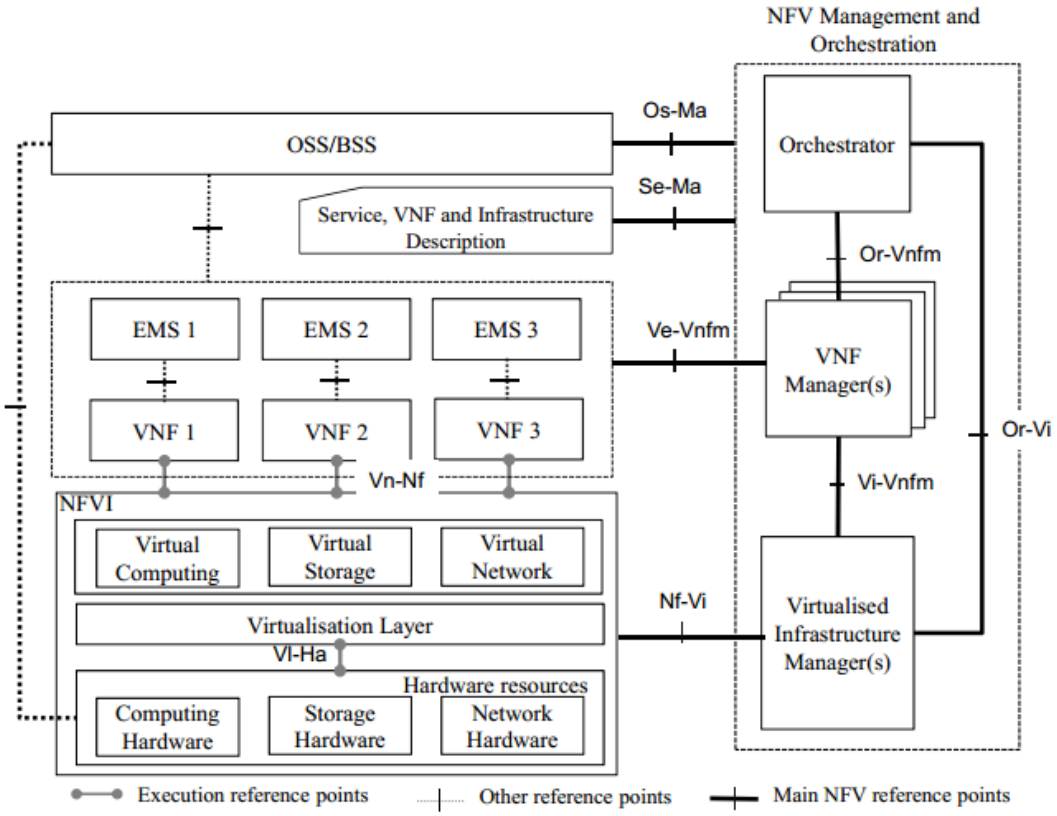

Figure 3.11: NFV reference architectural framework [GSNFV002]

It is not the aim of this text to state that "*NFV ETSI architecture matches the NaaS service model framework*" but instead to identify several points of connection and similitudes that justify current and future NaaS and NFV interworking while composing E2E network services. The following table summarises such identified similitudes between the ETSI model and NaaS principles.

Concept/Principle	ETSI NVF reference architecture	NaaS service reference framework
Virtualisation paradigm	The ETSI architecture leans on the NFVI which is composed of virtualised resources where to allocate VNF instances.	Abstraction and virtualisation is one of the pillars of NaaS model (see Section 3.1.1).
VNF and E2E service	VNFs are software instances which can be nested (creating VNF forwarding graphs) to compose E2E Network services and place on top of the NFVI.	NaaS works on top of virtualised (or not) infrastructures to create atomic network services (comparable to VNF instances) and orchestrate these infrastructures to compose more complex E2E network services.
Service Orchestration	The NFV MANO orchestrator handles the lifecycle management (instantiation, scaling – out/in, and performance of network services. It also deals with resource management and authorisation of requests for NFVI resource allocation [IETF88].	The NaaS reference model manages and orchestrates atomic and complex network services. The NFV MANO architecture splits this role into two, separated functional modules (the NVF orchestrator to manage the lifecycle of services) and the VNF manager (wit handles the life cycle management of VNF instances).
NVF Orchestration	The NFV MANO VNF Manager handles the lifecycle management of VNF instances and the overall coordination and adaptation between the NFVI and the E/NMS [IETF88].	
Infrastructure control and management	By means of the NFV MANO VIM functional block, it is possible to control and manage the NFVI compute, storage and network resources and collect and forward performance measurements and events [IETF88].	The NaaS model also requires control of the network resources (either physical of virtualised) while providing network services. Monitoring tasks are also an important part of NaaS in order to (i) control the provided services performance and/or (ii) to provide OTT applications with monitoring information for their own specific purposes.

Table 3.4: Identified similitudes between the ETSI NFV and NaaS models

Thus, NaaS is aligned with the NFV principle of providing E2E network services based on virtualisation techniques, in order to make the best of the benefits presented in the following section.

3.4.4.4 *NFV Benefits for the GÉANT Community*

The benefits that NFVs bring to the GÉANT community of NRENs and their users depend on the role and type of services provided. As such, the nature of the VNF mechanisms employed by the community may vary.

During the last months, the number of NFV forums, Live Demos and events has exponentially increased proposing NFVs to be deployed in all segments of the E2E network service provisioning chain.

Within ETSI, several companies (such as: Telefonica, BT, AT&T, Verizon) are defining generic Network Functions (NF, such as: firewalling, load balancing, carrier-grade NAT, routing, switching, etc.), which can be run on virtual machines. Thus, one example of a possible use could be to providing firewall functionality to NREN clients. This is aligned with the functionality provided by NaaS, as NaaS is based on having NFs that are Virtual Network Functions/resources (VNF, such as a virtual machine providing routing functionality) and Physical Network Functions/resources (PNF, such as a physical dedicated router).

Here below it is provided just a sample of already available NFVs proposed by ETSI ISG that could be of the interest of NRENs as part of the services to be provided or part of their own systems to enhance service provisioning, automation of processes etc. and also save money while providing network services to their customers. The proposed use cases may apply to different network segments (see Table 3.5).

Cloud Use Cases	Mobile Use cases	Data Center Use Cases	Last Mile
NFV Infrastructure as a Service (IaaS) (NFVIaaS)	Virtualisation of the Mobile Core Network and IMS	Virtualisation of CDN	Virtualisation of the Home Environment
Virtual Network Functions as a Service (VNFaaS)	Virtualisation of Mobile Base Station		Fixed Access Network Functions Virtualisation
Service Chairs (VNF Forwarding Graphs			
Virtual Network Platform as a Service (VNPaaS)			

Table 3.5: NFV ISG use cases presented in July 2013. Source [GSNFV001]

Finally, there are also several pure business solutions and recent NFV Proof of concept initiatives proposed by industrial partners: For instance, in October 2014, during the ETSI NFV PoC Zone, 22 NFV-based Proofs of Concept were presented [NFV-POC]. In October 2015, “the world’s largest showcase of live NFV proofs of concept” is planned during the 15th annual Broadband World Forum [BROADBAND].

4 OpenNaaS: A Tool for Tailoring GÉANT Services to NRENs and End Users

OpenNaaS is an Open Source platform for the provisioning of network resources and orchestration of network services. It allows the deployment and automated configuration of dynamic network infrastructures, and defines a vendor-independent interface to access services provided by these resources [OpenNaaS]. JRA2T2 adopted OpenNaaS for the introduction of NaaS service offerings in GÉANT.

4.1 OpenNaaS as a Network as a Service Tool

4.1.1 Service Support, Management and Orchestration

JRA2 Task2 is **proposing** future network services, as well as **providing support** to existing services through cross-related actions collaborating with other GN3plus activities. In addition, the adoption of software-defined and virtualised-function network environments entails software playing a much larger role than in traditional networking and opens a wide range of added-value features to be integrated, while providing service support and management. The most relevant "*nice to have"* features include:

- Enabling the development of network service applications exploiting interfaces to other services, platforms and technologies.
- Guaranteeing NaaS-based service extensions as/when needed to deliver additional NaaS capabilities and features.
- Ensuring NaaS service compatibility and integration with other popular technologies (mainly SDN and NFV) and services (service composition/service chaining).
- Guaranteeing that service requirements can be mapped to the prototyping and deployment of software modules and extensions for the proposed PoCs.
- Including virtualisation and abstraction mechanisms to support heterogeneous network devices.
- Providing recursive delegation and differentiated access rights over managed resources to different stakeholders.
- Providing a variety of user interfaces to handle services and resources.

The OpenNaaS software platform has been chosen to work with by JRA2 Task 2. Table 4.1 summarises the match of OpenNaaS capabilities to the aforementioned "nice to have" feature list and justifies its suitability to fulfil network services development and implementation.

Added-Value Future Network Services List	OpenNaaS Software Platform Performance
Development of service applications	Adopts the Operation, Administration, Maintenance and Provisioning (OAMP) model while providing NaaS-based services on top of network infrastructures ensuring suitable service development.
Development of service extensions	Provides the tools to extend and add novel features to implemented network services, enabling enhancement.
Guarantee service compatibility	By means of REST APIs, service compatibility and integration with other existing network services (e.g. MDVPN, BoD) can be achieved either to compose a new service or to enhance specific features on the existing ones. Besides services, it also allows implemented interfaces to integrate with other platforms, such as SDN control platforms (currently ODL, FloodLight and RYU) and Cloud platforms (e.g. OpenStack).
Support of heterogeneous network infrastructures	Relies on virtualisation, abstraction and includes a lightweight abstracted operational model that decouples the actual vendor-specific details. This entails a solution flexible enough to accommodate different designs and orientations and fixed enough so common tools can be build and reused across plugins.
Proper future network service mapping	The architecture consists of a core framework composed of reusable software and implements the extensions and capabilities to develop services and match their specific features and characteristics. Upcoming OpenNaaS releases are expected to simplify the service development (lowering the skills required to program one's own services) and will also add flexibility to the design.
Access right levels and rights delegation	Management as a Service (MaaS) is being studied and linked to NaaS service developments, so that it enables to offer different right levels to stakeholders, taking into account the level of permitted control that each of them have on the service and the assigned resources.
Stakeholder interfaces	Despite the fact it is not explicitly a part of OpenNaaS architecture, smart GUIs have been developed to enable service consumers (NOCs, NRENs, university users) with a proper interface presenting the options they have enabled according to the rights granted. OpenNaaS takes into account the users' experience of past projects and use cases to enhance service usability and performance.

Table 4.1: OpenNaaS software platform matching future NaaS service requirements

Thus, OpenNaaS is a tool to **develop** future network services, to **enhance/extend** current services (providing additional control over resources, enhancing their lifecycle by automatic specific processes, etc.) and to **integrate** with other technologies. Furthermore, OpenNaaS is currently experiencing a renovation of the core architecture. OpenNaaS developers are continuously evolving the tool, releasing new versions matching users' requirements and expectations. The following subsections provide an overview of OpenNaaS, the evolution of its architecture based on identified limitations, and the future of the tool.

4.1.2 Where did OpenNaaS come from?

OpenNaaS was developed in the European FP7 project MANTYCHORE in cooperation with European NRENs (HEAnet, NORDUnet). It is based on the Infrastructure as a Service Tool, but also has its roots in the privately funded MANTICORE II, whose functionality was added to the platform. MANTICORE II, in turn, stems from MANTICORE; the tool developed in these two projects was called Argia.

4.1.3 OpenNaaS Identified Limitations

Following the realisation that the old OpenNaaS architecture model (see Appendix B) was not optimal and did not address important stakeholder requirements, an analysis of the architecture of the platform was carried out to obtain a vision of where the further development of OpenNaaS should be headed. This was based on the feedback from partners in the Mantychore project, but also from users and developers of OpenNaaS. The analysis of the previous architecture included a deep compilation and understanding of the identified limitations while adopting OpenNaaS in pre-operational environments. The most important insights found were:

- The architecture of the core has been too complex and inflexible. When adding functionality to the platform, developers have to adhere to a predefined complicated structure and automated processes are not provided. Development is therefore costly.
- The finest functional granularity available is the capability, a concept grouping of several services offered by a device. Due to this granularity limitation, the amount of management that the core can offer is severely limited.
- The mapping between services and resources is static. Since it cannot be changed dynamically, the platform needs to be restarted whenever new functionality is added.
- The transactional model does not span multiple devices. Complex configurations carried out across several devices can therefore not be rolled back.
- The permission system is very limited. Role-based permission can only be assigned to services represented in the API and has to be configured manually.
- The organisational scheme of the platform needs to be reversed. Instead of the core depending on individual features, features need to depend on the core to allow for more flexible deployment.

4.1.4 OpenNaaS Evolution: How does the community overcome the limitations?

The actions driven to overcome previous limitations focus on two main areas: (i) Minimise the time necessary to deploy services and extend features and (ii) simplify and enrich the deployment possibilities.

To minimise the time necessary to deploying a new feature and to enrich the management possibilities of the platform, the core was completely redesigned using the main existing core concepts, resource and capability and adding a new concept, the service. This redesign includes:

- A finer functional granularity, namely, the service concept, is used within the core. Based on the service being the atomic unit of functionality, a series of additional management functions can be offered.
- A new transactional model allows for service execution across device and network borders.

- Built-in monitoring capabilities. Based on an observational model rooted in the platform's execution service, all events can be monitored and reacted upon. This mechanism facilitates simple tasks such as logging or auditing, but also allows formulating reactions to specific dynamic states of the platform.
- Dynamic mapping between resources and services. Instead of defining this relation statically, services and resources can be related dynamically. True 24/7 becomes possible.
- The interfaces a developer needs to implement to deploy new functionality are simplified and enhanced. All automated processing is taken over by the platform, without limiting the development possibilities. Additionally, libraries to access devices in a unified manner are provided. This reduces the necessary boiler plate code to a minimum.

To simplify and enrich the deployment possibilities, the core, its features and external functionality are as loosely coupled as possible. This allows:

- Full control over the inner core's functionality by implementing its control interfaces in a client specific manner. This includes control over the permissions, authentication and authorisation concepts used for the platform.
- Adaptation of the core's resource tree and the capabilities offered by these resources to tailor the platform exactly to the client's needs.

4.1.5 OpenNaaS Evolution – The New Platform Core

The two basic concepts used in OpenNaaS are the resource and the service. In the revised design, the service replaces the capability as the finest granularity of functionality managed by the platform. Capabilities, the third basic concept, represent groups of services. Capabilities are "bound" to resources, as seen in Figure 4.1 (A). The core has 10 capabilities, all of which are bound to it. Internally, capabilities may have logical dependencies, e.g. the Execution service logically depends on the Authorisation Control and the Monitoring Service on the Notification Service. Based on these three core concepts, OpenNaaS offers its functionalities, e.g. management of bindings between capabilities and resources, permission delegation on a service level or automatic API publication of a resource's services.

In addition to these three concepts, OpenNaaS internally manages a tree structure of resources. Each resource may, within its capabilities, manage sub-resources, for example, in Figure 4.1 (B), the core manages a set of routers using its Resource Management Capability and each of the routers in turn manage their interfaces using their Interface Management Capability.

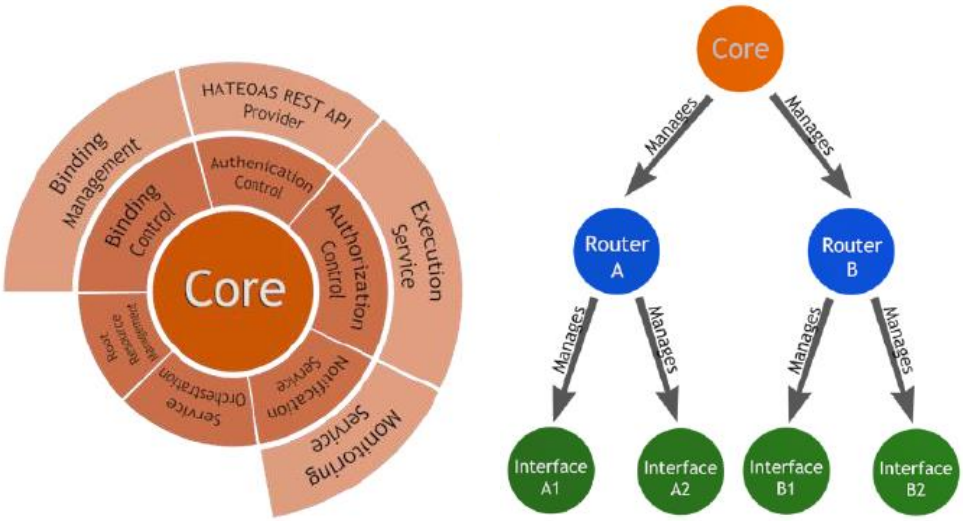

Figure 4.1: Novel OpenNaaS design. (A): The core resource and its capabilities. (B): Open NaaS's resource tree

The New OpenNaaS platform structure brings important advantages. Seen from a developer's point of view, the main advantage is:

- Using the automation mechanisms and libraries provided by the new core of OpenNaaS, there is a lot less code necessary, especially less boiler plate code. This increases maintainability and allows focusing on implementing application logic.

Seen from the client point of view, the two main advantages are:

- By implementing a few core interfaces of OpenNaaS the core's features can be fully controlled by logic specific to the client's institution, e.g. established security concepts can be integrated, authentication and authorisation can be adapted and binding between existing capabilities and resources can be controlled.
- The core concepts implemented allow developing, deploying and maintaining applications in a faster and more dynamic, flexible and adaptable way. This means less time to market while maintaining highquality levels.

4.1.6 Future Plans for the OpenNaaS Tool

The implementation of the full stack under the revised architecture is under development. Work on the GUI, the transaction model and the permission delegation system is being completed. In parallel, the use case of creating a virtual infrastructure is due to be finished by the end of GN3plus, for example, the platform is supporting the slicing of resources and their virtualisation.

Future platform development will concentrate on adding new features on top of the core and on broadening existing functionality. Specifically, a service orchestration layer will be implemented, allowing the creation of complex services using atomic services provided by the resources of the platform. The virtualisation functionality will also be extended to allow slicing of OpenFlow-enabled resources. The OpenNaaS community plans to go ahead with fostering collaboration within the GÉANT community. The integration with Cloud management

platforms, SDN-based platforms and other networking solutions is also in the scope of OpenNaaS development to ensure that the most up-to-date networking technologies are taken into account.

4.1.7 OpenNaaS Communication Channels (All)

OpenNaaS was established in the FP7 Mantychore project, but the intent has always been that the software should outlive a time-bound project [Mantychore]. Therefore, OpenNaaS was set up as an open source project that could attract users within and outside the project that incubated it, and continue beyond it.

The main interaction takes place on two lists: *OpenNaaS-users*, for users of the project, and *OpenNaaS-dev*, for discussion of the development between the developers team at i2CAT, other developers, and users. The development team monitors and responds promptly on both lists.

Plans are underway for the former Mantychore partners to set up a formal community to manage the future direction of OpenNaaS.

4.1.8 OpenNaaS Collaborative Work and Experiences in JRA2T2: the Tool Created by NRENs to Shape NREN Services

The different GN3plus JRA2T2 partners have provided their own experiences, while collaborating to extend the OpenNaaS tool by different means and channels. A short document that summaries such experiences can be found in Appendix A.

5 Coordination and Dissemination Activities

5.1 Coordination with Other GN3plus Activities

During GN3plus JRA2T2 partners are carrying out different actions to reach beyond the NaaS research activity and touch upon issues observed in SA services being addressed by NaaS concepts. In October 2013, a cross-activity session between SA3T1 (Bandwidth on Demand (BoD) Multi-Domain Service), SA3T3 (Multi-Point Virtual Private Network Service (MPVPN)) and JRA2T2 took place. The main target was to present SA services to NaaS partners so that technical requirements to enhance current services' performance would be identified. The following cross-task working items were proposed for Service Activity 3 and Networking Activity 1.

5.1.1 SA3 T1 – BoD

How can we extend BoD to the final user/consumer?

One of the major concerns of this service is the need to make end-users participants in the services they receive. BoD is a layer 2 inter-domain provisioning service (see Section 2.4) provided among NRENs. It would definitely be desirable to be able to extend the service to the end users.

How could we model a template to make BoD easy to be accessed and configurable by end users?

Similar to the templates created for the vCPE use case, it was proposed to provide a front-end with easy-to-fill-in templates, so that end users could activate the BoD service between the two ends of the communication by simply entering data into some fields in a template. The NaaS system working on top of the BoD service would take the responsibility to ensure that the service is properly configured.

5.1.2 SA3 T3 – MDVPN

Automation of Configuration Capabilities

JRA2T2 partners identified some aspects of the MDVPN service as being too manual and lacking automation while provisioning the service. JRA2T2 offered to analyse the service and identify potential points of the service workflow that could acquire more dynamicity to make service providers' lives easier.

This work item was not followed up as automation has not been among the priorities of the MDVPN roadmap.

5.1.3 NA1 T7 – NaaS and OpenNaaS Webinar and Training

In collaboration with this task, JRA2T2 plans to organise a webinar on NaaS principles and a tutorial/training on OpenNaaS tool in order to bring it closer to users and engage new users and developers in the community.

5.2 Dissemination Activities

Based on the dissemination plan developed to create NaaS awareness, the JRA2T2 team attended a number of events presenting its most relevant outcomes. The most relevant conferences and events in which the work done and NaaS-based demonstrators have been presented (or will be presented during the next few months) are listed below:

- TERENA Networking Conference (TNC2014): “Making Overlay Networks Simple with Networks as a Service” https://tnc2014.terena.org/core/presentation/7.
- NORDUnet Conference 2014: Speech on NaaS use case Network on Demand: "Automatic Provisioning of Networking Resources in Foreign Domain".
- http://events.nordu.net/display/NORDU2014/Automatic+Provisioning+of+Networking+Resources+in+Foreign+Domain.
- DeIC Conference 2014: Lightning Talk and Poster Presentation: "Network on Demand".
- http://www.deic.dk/program_tirsdag_2014.
- The Networking Conference (TNC2015): “Leveraging Software-Defined Networking for Information-Centric Video Distribution” (submitted).
- IEEE Conference on Network Softwarization (NetSoft 2015) (planned).
- Green routing (GreenNet) PoC demonstrator performance. Demo in SuperComputing (SC 2014).
- DCadmin PoC demonstrator performance. Demo in The 3rd workshop on Network Infrastructure Services as part of Cloud Computing (NetCloud 2013).
- http://new.opennaas.org/wp-content/uploads/2013/12/NetCloud13_ONManagement_FinalVersion.pdf.
- DCadmin PoC demonstrator performance. Demo in the Third International Conference on Cloud Networking (CloudNet 2014).
- The Networking Conference (TNC2015): “Choosing an open source network controller for a data center network“ (submitted).
- NaaS and OpenNaaS webinar and training (two sessions programmed in collaboration with NA1-T7 to take place in February 2015).

6 Conclusions

The main target of JRA2T2 is to define new NaaS-based service capabilities for NRENs and their user communities. To meet this goal, three main items have been addressed: (i) definition of NaaS-based use cases, to identify those suitable to be turned into PoCs and services, (ii) addressing the research aspects of NaaS and (iii) identification of NaaS service requirements that add value to the NaaS services proposition. To date, the major task achievements are:

- **Identification and analysis relevant to NaaS research areas**. SDN, NFV and MaaS were identified as prominent paradigms. NFV, SDN and NaaS are complementary in terms of the control they can add to network functions and services.
- **Identification of current limitations of service providers, as well as providing concrete solutions**. Use-case templates have enabled partners to identify specific limitations experienced by service providers. Use cases have focused on specific contributions where NaaS can be applied, i.e. focus on a small-scale, specific problems and their solution.

  Major outcomes of this work include the PoC and demonstrators that have been (or will be) presented in different reference networking events.
- **JRA2T2 adopted OpenNaaS as the network management software platform to implement the use cases and develop the PoCs**. This has enabled prototype extensions to the platform and increased participation of the OpenNaaS community. More importantly, the JRA2T2 partners have provided precious feedback on the limitations presented by OpenNaaS. This has enabled a re-thinking of the model, and provides a more flexible and versatile tool design that is appropriately matched to the requirements stakeholders may impose for their services.

The following specific NaaS actions are planned for completion before the end of the project (April 2015):

- **Create a real value proposition**: State what makes NaaS useful to users and report on the identified limitations/needs/specific requirements foreseen in NRENs' facilities while managing their services.
- **Work to acquire user commitment**: It is important to identify the users that are currently not served. Validation of the services with the user will be key. While the users may be interested in the technology, they will certainly be interested on the business benefits as well (regardless of the ROI). It is fundamental to keep users motivated and active throughout development of the service.
- **Provide clear statements defining NaaS, SDN and NFV** and their limitations of use, to drive the vision and goals, and also communications with potential users.
- **Clarify the complementarities and advantages of OpenNaaS, OpenDaylight and other SDN platforms**. As an example, there is no OpenNaaS controller, since that is not the goal for which it was conceived. Nevertheless, OpenNaaS can act as the service and NFVs orchestrator while OpenDaylight could be used for control plane purposes.
- Provide valuable NaaS capabilities for existing use cases and new, proposed ones. Some examples include:
  - vCPE on its own has no use case for NaaS, but vCPE with Network-on-Demand (or Network to the lab or Science DMZ [SciDMZ]) would make sense, as it could solve the immediate problem of extending specialised connectivity to the last mile.

- Enhance automation processes in Multi-Domain Virtual Private Network (MDVPN) GÉANT services based on the NaaS approach.
- INaaS vs SDN to extend the network into the DC is also an interesting use case.

6.1 Next Steps

As a result of outputs of JRA2T2, there are recommendations for the future direction of NaaS-based work (in order of priority and within the next 12 months).

1. Design novel network services built on NaaS principles, VNFs and SDN apps:
   - Integration of NaaS management and SDN control platforms to enable the development and tailoring of new network services by making use of virtualised resources, abstractions and software constructs, and enable managed, end-to-end network provisioning services. Although NFV deployment has been localised, it has great potential in multi-domain and federated scenarios, and will complement NaaS and SDN to build granular and innovative network applications at different levels of abstraction.
2. Exploit the Use Cases and the PoCs delivered within GN3plus within service development processes to take NaaS-based service capabilities to the market as needed.
3. Use the new OpenNaaS tool design to improve coordination between other technologies (e.g. SDN, NFV).
4. Continue to provide information about NaaS-related topics to the relevant SA Activities (especially SA3 and SA4).
5. Ongoing dissemination of NaaS and further service development. Continue positive communications with partners to support dissemination strategies.

Appendix A Validation of NaaS Findings: Use Cases and Requirements

An initial set of use cases was proposed to trigger NaaS discussion at the beginning of JRA2T2 (see MJ221 [MJ221] document). Later, additional use cases have been incorporated to the task for its complete analysis and to retrieve a complete list of functional requirements that should be taken into account while developing the use case. The analysed use cases (see Figure A.1) are:

- Virtual CPE – vCPE
- IP Network Overlay – IPNO
- Data Center Administration – DCAdmin
- Network on Demand – NoD
- Energy Description model – GreenNet

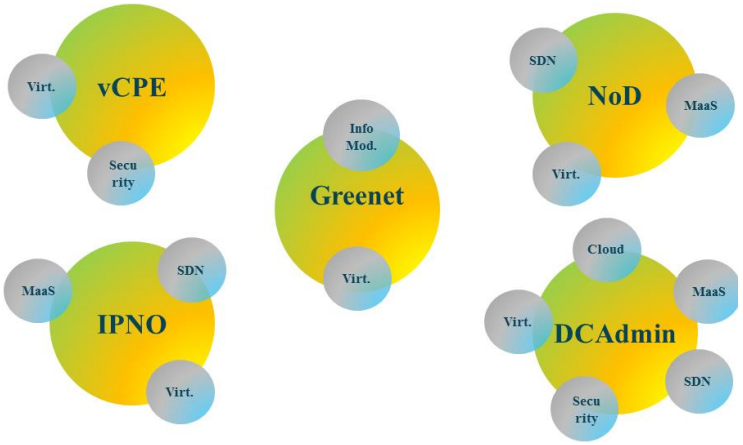

Figure A.1: Analysed use cases and their connection to JRA2T2 research areas

For each of the use cases, the following items have been addressed:

- Basic description
- Main goals
- Scenario
- Stakeholders
- User stories and flowchart
- Functional requirements template
- Conclusions

The use cases were based on a template specifically developed for the task (see Figure A.2). Since they were partially described in a previous milestone document, the full analysis of the use cases can be found in Appendix B.

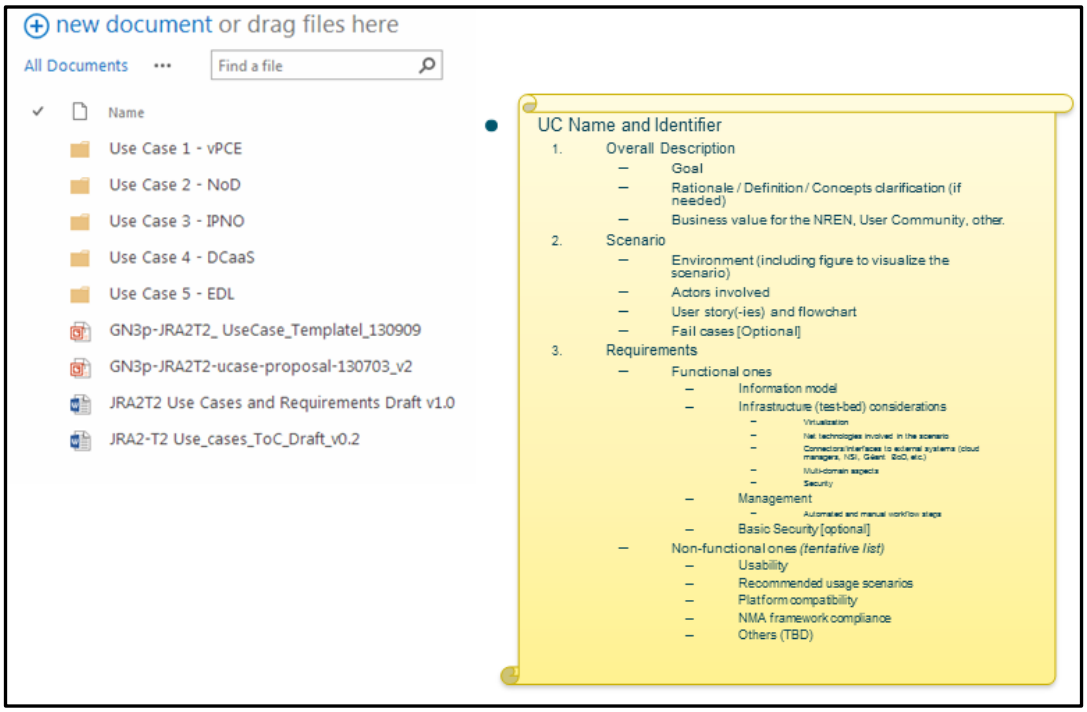

Figure A.2: GÉANT JRA2T2 Use case repository and sample of the fulfilled template for the analysis

Appendix B Demonstrators and PoCs

In this appendix, the full description of the demonstrators and Proofs of Concept (PoCs) that have been developed are provided, making use of the OpenNaaS management software platform, and shown at different events and conferences (see Section 5.2). Some PoCs have started from the definition of the use cases presented in Appendix A and others were proposed later as team member proposals. Developed proofs of concept include:

- Data Centre administration (based on the use case of the same name).
- GreenNet PoC (also based on the use case of the same name).
- NaaS ICN (Information Centric Network) over SDN service.

In the next section, we present the Proofs of Concept.

B.1 Data Centre Administration

B.1.1 Description of the PoC

The PoC of the Data Centre (DC) Administration use case is focused on the use of OpenNaaS as a platform to provide user-friendly, but extensive, access to a Data Centre's managed resources, through a web portal that can be used by non-network administrative users.

Since the Data Centre use case has a lot of requirements, it was decided to focus on a few "must haves". These are:

- Support DC-related capabilities for Juniper EX series Ethernet switches.
- Graphical interface for managing resources / capabilities.
- Support topology discovery (using LLDP) for maintaining the DC topology.
- Support centralised VLAN database (as described in use case), validity checking for trunks.
- Support additional vendors (e.g. Cisco/Arista) as an action set for the router resource, featuring DC-related capabilities e.g. using the Arista JSON API or NETCONF.

From these requirements, the ones that were easiest to implement and that provided the most (new) functionality were selected and implemented as far as possible to support the PoC. At the time of writing, the first three items have been implemented and demonstrated.

B.1.2 Business Value

The main business value resulting from this PoC lies in the reduction of the administrative load for network staff by delegating small operational tasks to the system administrators of the server equipment. The system administrators will have a better understanding of the network infrastructure and will have more freedom to connect their server equipment on the access layer. The administrative freedom and increased insight into the network will also potentially lead to savings in operational expenditure.

B.1.3 Implementation Details

B.1.3.1 *Used Technologies*

Initial work on the Data Centre use case PoC began in January 2014. The first versions were based on PHP. After the summer of 2014, it came apparent that there was a better approach to build the application, and it was decided to rewrite the code as a Python WSGI application. WSGI [WSGI] is a standard that allows universal communication between web applications and servers, which can be seen as an improved alternative for CGI (Common Gateway Interface). There are a number of web application frameworks making use of WSGI; Django and Flask are two frameworks that are used broadly. For the PoC it was decided to use Flask.

Flask supports many features that allow easy and efficient coding. Some of the main features that are used are: URL routing (using Werkzeug), HTML templates (using Jinja2), and stand-alone web application support (without using an additional webserver). The details of these features can be found in the online Flask documentation, and will not be explained in detail here [FLASK].

B.1.3.2 *Application Design*

The WSGI application has two roles in the design:

- It handles the communication with the REST interface of OpenNaaS. This is done using a separate python module containing a set of python functions, which creates requests and parses responses from and to the OpenNaaS REST interface. This can be seen as the middleware layer of the application.
- It provides the front end to the user to operate the WSGI application. This is mostly done using the Flask framework and some static HTML and JavaScript pages.

Using JavaScript, it is possible to send AJAX calls directly towards the REST interface of OpenNaaS. In this sense it would not be necessary to have any middleware layer in between the GUI and OpenNaaS. However this is not a favourable approach, because this means that the users' web browser communicates directly with the OpenNaaS REST interface. Instead, our implementation sends JSON calls to the WSGI application, which in turn, generates calls to the REST interface.

This offers better security, since in this way the WSGI application acts as a proxy. Requests towards the REST interface are only sent through the WSGI application. Also, the WSGI application consolidates requests and responses from the REST interface and sends abstracted JSON data back to the web browser, which allows a more efficient way of handling the data generated by the REST interface.

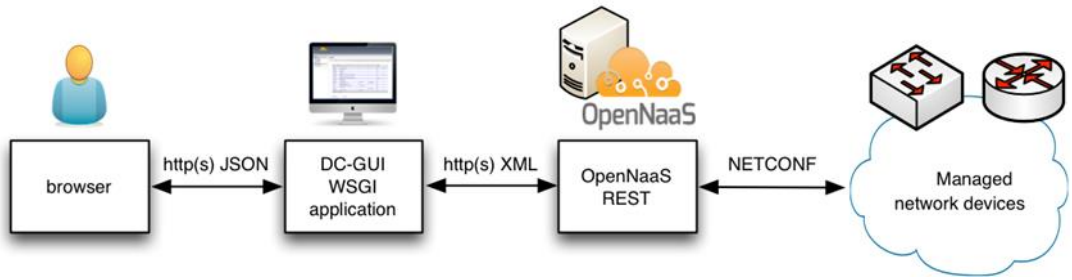

Figure B.1: Datacentre GUI administration information flow

B.1.3.3 *Frontend*

The frontend of the WSGI application is heavily based on widely-used web technologies such as CSS, JQuery and AJAX.

To keep the application simple to develop, static HTML templates are used to generate the GUI elements (e.g. tables, buttons, and checkboxes.) The interaction between the graphical elements is handled in a single JavaScript module. The same module also handles the AJAX requests towards the WSGI interface, which in turn handles the REST calls to/from OpenNaaS. This has proven to be an efficient design that clearly divides the presentation and middleware layers of the application.

B.1.4 Future Work

The current state of the PoC is that it is working well as a demo, but does not have enough features to implement it in an enterprise environment, such as SURFsara.

The possibilities to improve and add functionality are virtually endless. There is still a long list of outstanding requirements, of which some can be prioritised.

One focus would be to enable configuration of managed resources, not from a single device perspective, but instead, to define networks which allow configuration of multiple resources at the same time. For instance, a VLAN could be defined and applied to a group of resources at the same time as its creation.

Recently an initial topology discovery function has been implemented, but this could be improved. Some other "must haves" include multiple vendor support, user authentication and authorisation, monitoring of device parameters, and abstraction of VLAN information gathered using topology discovery.

B.2 GreenNet Service

B.2.1 Description of the PoC

The PoC of the GreenNet use case proves that OpenNaaS can be used as an energy-aware platform, which provides energy monitoring and energy-aware routing to its managed networks. Through web portals, users and providers can monitor the energy state of resources, and the providers can also create for the users routing paths that take into account energy consumption.

Implementation

We have implemented three energy-aware capabilities in the OpenNaaS, as following:

- Energy description capability: create and manage information about energy source, power meters, green metric, and power state, etc.

- Energy monitoring capability: obtain data in the observed metrics (power, energy, CO2 emission rate, electricity price) and the calculated metrics (total Electricity, CO2 emission, energy efficiency).
- Green routing capability: find the “greenest” routing path in terms of power, cost and emission metrics and use the flow control capability for OpenFlow switches in OpenNaaS to create flow tables in OpenFlow controllers.

A Mininet [Mininet] network using OpenFlow controllers to control network flows has been used in Green Net. The real-time monitored power consumption of cluster nodes in University of Amsterdam have been used to mimic the power consumption of the Mininet network. According to our experience on power monitoring, the power consumption of active network devices is almost stable, and barely depends on the throughput of traffic. The nodes we choose also have similar power characteristics.

A GUI client, named “Green Routing Demo” has been developed to show the feasibility of the capabilities. The clients provide the web portals to easily use these capabilities. The clients are written in Java and JavaScript.

As seen in Figure B.2 we focus on a simple network topology. The client demonstrates three functions:

- Show the real-time power information of each node.
- Find a green routing path in terms of three green metrics when given start and destination hosts.
- Show the power information of each specified routing path.

B.2.2 Benefits of GreenNet

On the whole, the GreenNet use case improves the application scope of OpenNaaS in terms of cost and sustainability. It is possible to reduce the energy cost and Greenhouse Gas (GHG) emissions of a network infrastructure. Both network users and network providers can use OpenNaaS to understand energy, cost and sustainability information of network resources. Besides this, it also lays the foundation of supporting the users’ request for resources or a subset of ‘green’ (more energy efficient) networks. Also, other energy-aware network management capabilities can be implemented based on our work.

B.2.3 Future Work

The green-routing mechanism in the demo is simple. When an OpenNaaS user sends a routing path request, OpenNaaS calculates a ‘green’ routing path, optimizing the energy consumption. This is a locally optimal solution. If multiple requests arrive, a global solution to calculating green routing paths is needed. A global approach to green routing has been added to the longer-term view.

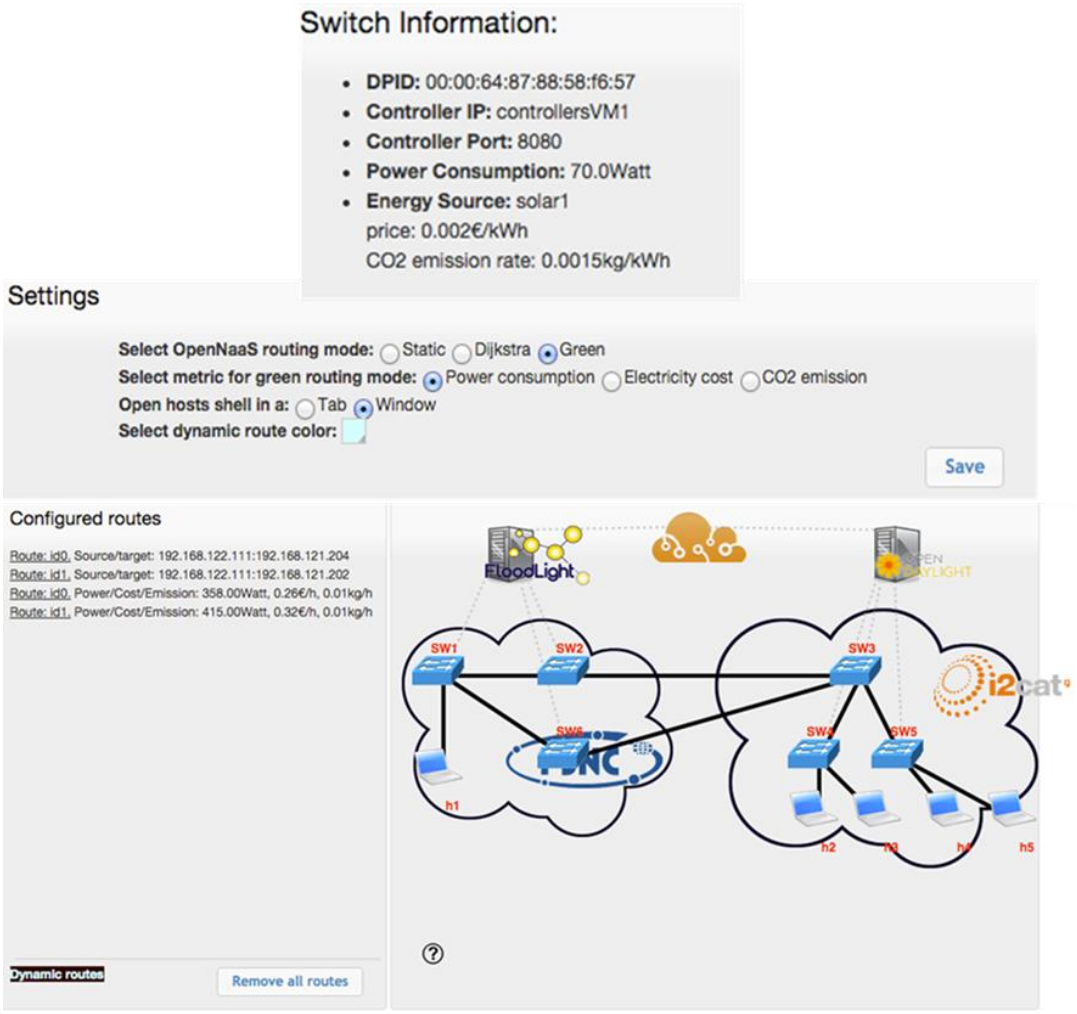

Figure B.2: Three functions in the GreenNet demonstration

B.3 ICN over SDN service

B.3.1 Description

This PoC responds to the needs of NREN customers that want to distribute content among their peers in the research and education community. We extend the reach of NaaS to also include 'Content Distribution Networks as a Service' as an additional service in the GÉANT portfolio. Such distribution is performed by means of an ICN that is conveniently and easily set up through an OpenNaaS web interface. The provisioned network operator has full control of content and traffic handling over the ICN and can provide content consumers with the requested resources based on their name and not on communication endpoints. This is implemented through an SDN application that distributes content throughout a network of caches and keeps track of the location of resources. The OpenFlow protocol is leveraged to steer flows to the appropriate cache or destination.

The provisioning flow to create the ICN service is shown in Figure B.3. For demonstration purposes, video distribution is used as an example, as it is the most demanding case where the greatest benefit can be obtained

by using such ICN service (higher bandwidth savings within the NREN and increased Quality of Experience (QoE) for the user). This PoC is carried out over the SDN that the team has deployed and set up within the University of Murcia.

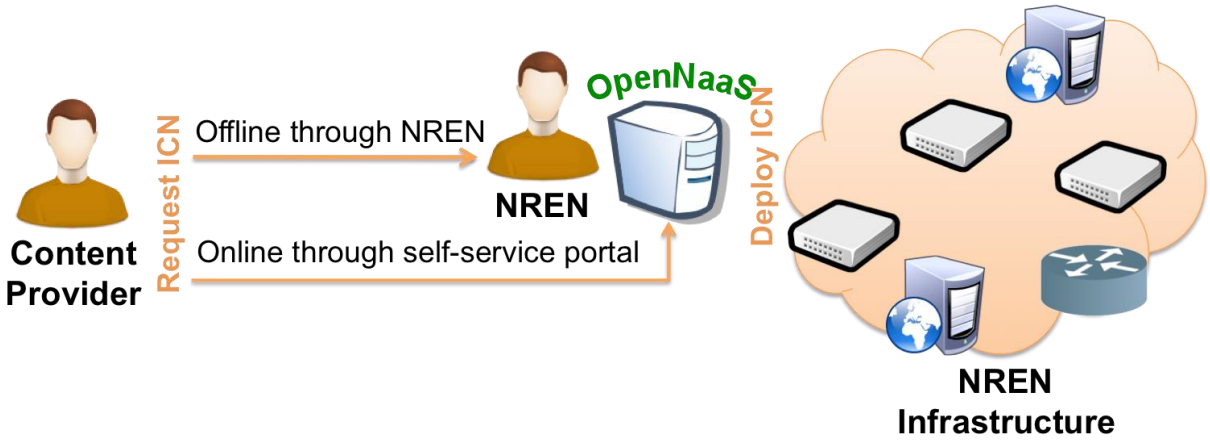

Figure B.3: Provisioning workflow for the ICN over SDN service

B.3.2 Business Value

The **content producer** (NREN customer) can provide its peers (other NREN customers) with an increased QoE. The solution demonstrated in this PoC is flexible enough to allow for paid content to be distributed only to the intended subscribers. Using the ICN service offered by the NREN is superior to other alternatives, which have these disadvantages:

- The content producer scales up its own infrastructure (server capacity, network bandwidth); Increases capex and opex.
- The content producer hires a third-party CDN provider: The trust relationship is already established with the NREN, another one with the CDN provider is required; content might not abide by local law; content is typically farther away from the consumers, potentially decreasing the QoE; a paid content business case might be more difficult to realise.

The **NREN** incorporates another service into its NaaS portfolio, adding an additional source of revenue. In addition, the ICN allows for bandwidth savings in the NREN core by pushing content near the consumers at the edge of the network. This benefit is higher with bandwidth-demanding applications such as video streaming.

The **content consumer** (NREN customer) can access the content generated by the producer (another NREN customer) from a close location, therefore achieving high QoE.

B.3.3 Assumptions and Requirements

The ICN service demonstrated within this PoC meets the following criteria:

- Content is delivered over HTTP, currently the most common distribution transport for video streaming.

- The NREN infrastructure is SDN-enabled.
- The service must be completely transparent to content producers and consumers without requiring new software installation or configuration in the corresponding infrastructure.
- The service must be provisioned through OpenNaaS in a simple and intuitive way.

B.3.4 Architecture

The ICN service consists of the following functional elements (Figure B.4).

- **OpenNaaS**. The ICN service is provisioned through OpenNaaS. This can be accomplished either online by providing NREN end users with direct access to a portal, or offline by contacting NREN staff through traditional communication channels. OpenNaaS instantiates the ICN by leveraging the ICN Northbound API provided by the ICN Application.
- **ICN Application**. The ICN Application exposes the ICN service through an **ICN Northbound API**. It keeps a global view of each ICN and decides on the network cache(s) where every piece of content to be distributed is stored. Such an application is implemented on top of an SDN Controller, so that it can steer HTTP requests to the appropriate cache where the named resource is hosted. In order to realise traffic steering, it offers a **Private Northbound API** to ICN Proxies that indicate what resource is being requested.
- **SDN Controller**. The SDN Controller is in charge of handling the NREN infrastructure through an **SDN Southbound protocol**. It keeps a global view of the network and allows the ICN Application to program network devices with the required network flows.
- **ICN Caches**. Web caches that host HTTP resources. They are located at the edge of the network close to content consumers.
- **ICN Proxies**. ICN service-aware HTTP proxies that notify the ICN Application about the HTTP resources that are being requested by ICN consumers.

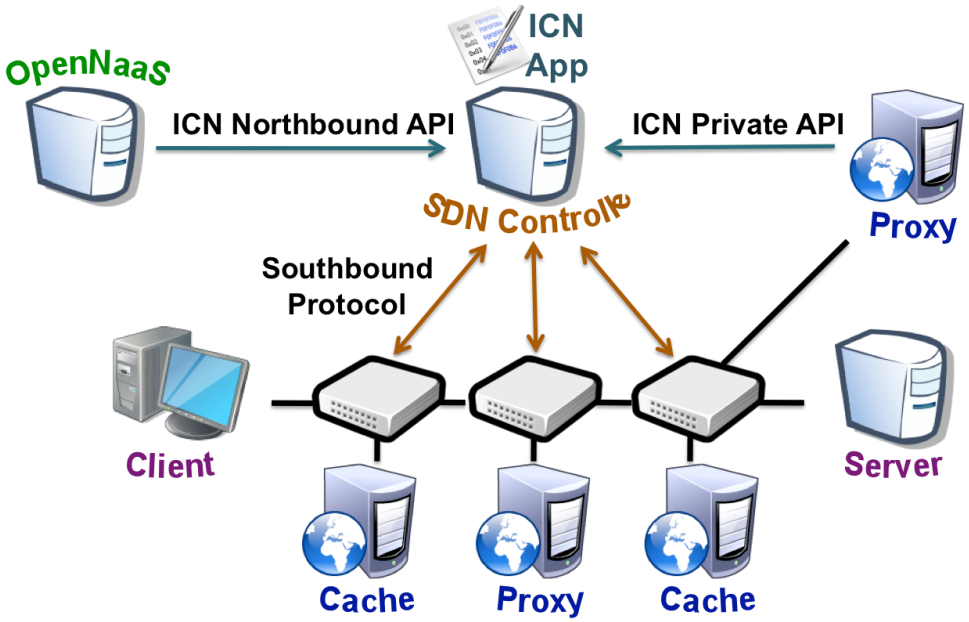

Figure B.4: Architecture of the ICN over SDN service

B.3.5 Implementation Details

OpenNaaS is being extended with an additional capability that leverages the ICN Northbound API of the ICN application.

The SDN Controller of choice for this PoC is **Floodlight**. The rationale for this decision is twofold: i) the OpenNaaS team has previous experience with this controller, and ii) it provides an stable API with enough functionality to demonstrate the ICN service concept (other more future-proof options such as OpenDaylight were under heavy development when this work started). Floodlight is an **OpenFlow** controller, so we use this protocol at the southbound interface between the controller and the network devices.

The ICN Application has been developed as a **Java Floodlight module**. This application exposes **RESTful Northbound APIs** for both the OpenNaaS client and the ICN Proxies. The Public Northbound API which is used by OpenNaaS allows creating, retrieving, updating and deleting different objects (Figure B.5). It is employed to create as many ICNs as required, register the proxies deployed within the network, associate different content providers to each ICN, and allocate caches in a per-ICN fashion. In addition, it provides information about the different resources which have been requested, the caches in which they have been hosted, and their popularity (number of requests).

URI	GET	POST	PUT	DELETE
/cdn	✓	✓	✗	✗
/cdn/{name}	✓	✗	✓	✓
/cdn/{name}/provider	✓	✓	✗	✗
/cdn/{name}/provider/{name}	✓	✗	✓	✓
/cdn/{name}/cache	✓	✓	✗	✗
/cdn/{name}/cache/{name}	✓	✗	✓	✓
/cdn/{name}/resource	✓	✗	✗	✗
/cdn/{name}/resource/{id}	✓	✗	✗	✗
/cdnproxy	✓	✓	✗	✗
/cdnproxy/{name}	✓	✗	✓	✓

Figure B.5: Public RESTful API exposed by ICN Application. Base URI is /wm/cdnmanager/v1.0

We have developed a **Python transparent HTTP proxy** that uses the Private Northbound API exposed by the ICN Application. In this way, the ICN Application notifies the URI of the requests that are being issued by the content consumers and redirects the dataflow to the appropriate cache. Such a proxy is required to implement this service because the resource name of a request (URI in the HTTP header) is only known after the TCP three-way handshake. Therefore, the proxy performs delayed binding (aka TCP splicing) before notifying the ICN Application about the resource name and initiating the connection to the destination server (the network itself directs the connection to the cache determined by the ICN Application). The role of such a proxy is also appropriate for other scenarios not demonstrated in this PoC, including delivery over HTTPS and usage of ICN protocols within the NREN (e.g. CCNx) instead of an SDN application-based implementation.

Finally, Open Source **squid** web caches are employed as ICN Caches. No modifications are required to these elements.

After OpenNaaS registers the available ICN Proxies, ICNs, Caches and Content Providers, the workflow of this PoC is summarised in the following bullets:

- The ICN Application programs network devices to punt HTTP requests from consumers to the controller for further inspection. If a request is found to be tied to a content provider, the ICN Application programs a flow towards the best suited ICN Proxy. Other traffic flows follow the 'normal' path determined by the SDN Controller.
- Upon processing a request, the ICN Proxy uses the ICN Private Northbound API to notify the ICN Application about the requested resource.
- The ICN Application checks whether the resource belongs to any established ICN. If this is not the case, the flow from the proxy is forwarded towards the intended destination following the path established by the SDN Controller. Otherwise, if the resource is not hosted in an ICN Cache yet, the ICN Application chooses a cache to host the content according to the NREN policy.[1] A path from the proxy to the cache is established by means of OpenFlow.
    - When a cache hit occurs, the content is forwarded to the proxy (and then to the consumer) through the paths established by the SDN Controller. In the case of a cache miss, the content is first downloaded from the provider following the path established by the SDN Controller, and then delivered to the consumer via the proxy.

B.3.6 Future Work

The software developed is flexible enough to easily accommodate further enhancements in future versions of this work. We expect to continue working on the ICN over SDN service as detailed in the following:

- Extend the ICN Northbound API to allow OpenNaaS to specify a subset of the network resources that are available for a given ICN. This helps to implement the paid content business case in which only subscribers of a given provider are allowed to retrieve resources (the network would enforce such policy).
- Given that the ICN Application centralises knowledge about the placement of all resources, efficient policies for content placement can be envisioned. We expect to evaluate different approaches in this regard.
- Enhance the ICN capability of OpenNaaS by allowing on-demand instantiation of ICN caches and proxies as virtual functions in private clouds (e.g. based on OpenStack).

[1] In this PoC, the closest cache from the consumer is chosen.

Appendix C Status of OpenNaaS Development

C.1 OpenNaaS

The NaaS model development has been brought forward with the OpenNaaS framework for easy and rapid prototyping and proof of distinct NaaS concepts. OpenNaaS is a robust and extensible open-source service and network control and management platform, which provides a suitable set of tools for the management and operation of network connectivity services. The software has been created to offer neutral stakeholders a common NaaS software stack, oriented for applications and services, to contribute to and benefit from. OpenNaaS is a resource and service management platform with the following characteristics:

- Secure, robust, and flexible open source framework for an agile and reliable resource and service management.
- Quick, hassle-free orchestration and composition of customised services and applications.
- Abstraction and virtualisation of resources, functions, and services to provide vendor-independent solutions.
- Open and accessible interfaces for building tailored visualisation platforms.
- Automation of processes for efficient service chaining and opex reduction.

The architecture is built around the resource and services concept. There are different, reusable building blocks, common to all the extensions and abstractions. In essence, a resource represents a manageable unit inside the NaaS concept. Each resource holds a list of capabilities; different actions that can be performed by each resource. OpenNaaS allows the creation of a software resource (e.g. Devices, Networks, Network Functions) and management of the offered services.

C.2 OpenNaaS Current Status and released version

The OpenNaaS current released version is 0.30. This release includes the following new implemented blocks of functionalities:

- Provide the ability of reading, but also reporting, network statistics by circuit, flow and device port to SLA managers and SDN applications.
- Controller Information capability has been introduced in order to obtain statistics on the memory usage and the health state of the OpenFlow switch controller.
- Enhancement of some existing features and capabilities from previous versions.

Further information on OpenNaaS 0.30 version can be found at [OPENNAAS030].

C.3 OpenNaaS Next expected release

A number of features have been developed as a result of GN3plus NaaS-based use cases. These have been scheduled as part of the next OpenNaaS release. Most relevant features relate to the so-called DCAdmin case, and include:

- Support of datacentre-related capabilities for Junos EX switches, in Karaf and REST.
- Implementation of a graphical interface to manage networking resources / capabilities (possibly REST).
- Support of topology discovery (using LLDP) for maintaining the datacenter topology.
- Support centralised VLAN database (as described in use case), validity checking for trunks.
- Make aggregated interfaces visible when created.
- Create a link aggregation from interfaces that already have a sub-interface specified or MTU set (or any other information).
- Improve VLAN options parsing and setting.

The next OpenNaaS release is also expected to incorporate critical features that will:

- Minimise the time necessary to deploy a new feature and to enrich the management possibilities of the platform.
- Simplify and enrich the deployment possibilities, the core, its features and external functionality are as loosely coupled as possible.
- Include the transaction model and the permissions delegation system.
- Enable slicing of OpenFlow resources.

The service orchestration layer will be complemented by enabling the creation of complex services using atomic services provided by the resources of the platform.

C.4 OpenNaaS Licensing Scheme

The core development team is part of Professional Services of the DANA department at i2CAT Foundation. OpenNaaS is driven by the needs of its stakeholders and users.

OpenNaaS software has two parts: core and extensions.

- Core is licensed under LGPLv3 licence, allowing commercial use, distribution, modification, patent grant, private use and sublicensing. Any modification of the core must be licensed with the same licence (or even GPLv3). Any application under any licence can make use of the core, it is only needed to notify the usage of OpenNaaS core as an LGPLv3 licensed software. Anyone can develop extensions on top of the core under any licence.
- Extensions provided by OpenNaaS are licensed under Apache v2 licence, allowing commercial use, distribution, modification, patent grant, private use and sublicensing. Any modification of an extension's source code must be explicitly noticed. Both licences guarantee copyright and patent rights to the software owner, i2CAT Foundation, under same licence terms for original source code and any modification of any contributor.

Component	Licence	Repository
OpenNaaS Core	GPLv3	https://github.com/dana-i2cat/opennaas
OpenNaaS Extensions	Apache 2	https://github.com/dana-i2cat/opennaas

Table C.1: Link to OpenNaaS core and extension repository

References

[4WARD] http://www.4ward-project.eu/index.php

[BluePlanet] https://www.cyaninc.com/products/blue-planet-sdn-platform/planet-view

[BROADBAND] http://broadbandworldforum.com/nfv-proof-of-concepts/

[CSC] CSC MaaS Custom Applications – Revolutionize Application Management http://assets1.csc.com/application_services/downloads/CSC_MaaS_Custom_Applications102913.pdf

[Dijkstra] Dijkstra, F., van Oudenaarde, B. et al., "A Terminology for Control Models at Optical Exchanges", *Inter-Domain Management: First International Conference on Autonomous Infrastructure, Management and Security (AIMS 2007),* Springer, Berlin Heidelberg, 2007.

[eTOM] Business Process Framework, Addendum D: Process Decompositions and Descriptions, Version 12.3, 2012.

[Fayazbakhsh et al] Fayazbakhsh, S. K., Reiter, M. K., & Sekar, V. (2013). "Verifiable network function outsourcing*", Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization - HotMiddlebox '13* (pp. 25–30). New York, New York, USA: ACM Press doi:10.1145/2535828.2535831

[Feamster] Feamster, N. (2010). "Outsourcing Home Network Security" *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks - HomeNets '10* (p. 37). New York, New York, USA: ACM Press. doi:10.1145/1851307.1851317

[Flask] Flask WSGI framework documentation: http://flask.pocoo.org/docs/0.10/

[Fujitsu] Fujitsu IT Management as a Service – Capability overview http://www.fujitsu.com/downloads/AU/pdfs/2p_bro_ITMaaS_overview_preview.pdf

[GESYSERS] http://www.geysers.eu/index.php/deliverables/,Deliverable D1.1, section 2.2

[Gibb et al.] Gibb, G., Zeng, H., & McKeown, N. (2012). Outsourcing network functionality. In *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12* (p. 73). New York, New York, USA: ACM Press. doi:10.1145/2342441.2342457

[GitHub] https://github.com/osrg/ryu/tree/master/ryu/app

[Gkantidis] Christos Gkantsidis, H. B. (2010). Network Management as a Service. Retrieved from http://research.microsoft.com/apps/pubs/default.aspx?id=132675

[GreenPages] http://www.greenpages.com/

[GSNFV001] http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs _nfv001v010101p.pdf

[GSNFV002] http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01 _60/gs_nfv002v010101p.pdf

[GSNFV003] http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.01.01_ 60/gs_nfv003v010101p.pdf

[GSNFV004] http://www.etsi.org/deliver/etsi_gs/nfv/001_099/004/01.01.01_ 60/gs_nfv004v010101p.pdf

[GSNFVPER002] http://www.etsi.org/deliver/etsi_gs/nfv-per/001_099/002/01.01.-01_60/gs_nfv-per002v010101p.pdf

[IETF88] Ersue, M. “ETSI NFV Management and Orchestration” http://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf

[ITU-T] ITU-T focus group – Cloud Technical report, Part 1: Introduction to the cloud ecosystem: definitions, taxonomies, use cases and high-level requirements, 2012, www.itu.int/en/ITU-T/focusgroups/cloud/Documents/FG-coud-technical-report.zip

[Jain13] http://www.slideshare.net/rjain51/m-17nfv

[LD43] http://www.ict-lightness.eu/wp-content/uploads/2014/06/lightness-D4.3_final.pdf

[Loggly] https://www.loggly.com/

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

[Microsoft-MaaS] Gkantsidis, C., Ballani, H., *Network Management as a Service*, Microsoft Research Technical Report, MSR-TR-2010-83, 2010.http://research.microsoft.com/pubs/132675/Network%20Management%20as%20a%20Service%20-%20MSR-TR-2010-83.pdf

[MJ221] García-Espín, J.A., (editor), et al., NaaS/NSI/Virtualisation Research Roadmap. Internal document to the GN3plus project (MJ2.2.1). August 2013.

[Mininet] http://mininet.org/

[NFV-POC] http://www.layer123.com/etsi-nfv-poc-zone

[NIST] Mell, P., and Grance, T., “The NIST definition of Cloud Computing: NIST Special Publication 800-145”. Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[NML] van der Ham,J., Dijkstra, F., Lapacz, R., and Zurawski, J., The Network Markup Language (NML) “A Standardized Network Topology Abstraction for Inter-domain and Cross-layer Network Applications”,*Proceedings of the 13th Terena Networking Conference*, 2013.

[NML-Schema] van der Ham, J., Dijkstra,F., R.Lapacz, R., and Zurawski, J. Network Markup Language Base Schema version 1. http://www.ogf.org/documents/GFD.206.pdf, 2013.

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

[ODLWI] https://wiki.opendaylight.org/view/OpenDaylight_Controller:REST_Reference_and_Authentication

[OESS] http://www.internet2.edu/products-services/advanced-networking/oess/

[OpenFlow] McKeown et al. (April 2008). "OpenFlow: Enabling innovation in campus networks". *ACM Communications Review*. Retrieved 2009-11-02.

[OpenNaaS] http://opennaas.org/

[OpenNaaS030] http://opennaas.org/2014/11/19/opennaas-0-30-released/

[ONF-NBI-Charter] Open Networking Foundation NorthBound Interface Working Group (NBI-WG) Charter. https://www.opennetworking.org/images/stories/downloads/working-groups/charter-nbi.pdf

[ONF-NFV] https://www.opennetworking.org/solution-brief-openflow-enabled-sdn-and-network-functions-virtualization

[ONF-SUB] http://www.onfsdninterfaces.org/

[ONOS-WP] Introducing ONOS - a SDN network operating system for Service Providers http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf

[OPNFV] https://www.opnfv.org/

[Pellet] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y.“Pellet: A Practical OWL-DL Reasoner”*Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[RFC 2741] Daniele, M., Wijnen, B., Waton, T.J., Ellison, M. Francisco, D. “Agent Extensibility (AgentX) Protocol Version 1” https://tools.ietf.org/html/rfc2741

[RFC 3746]	Yang, L., Dantu, R., Anderson, T., Gopal, R., “Forwarding and Control Element Separation (ForCEs) Framework”, April 2004, https://tools.ietf.org/html/rfc3746
[RYUBook]	http://osrg.github.io/ryu-book/en/html/
[SciDMZ]	https://fasterdata.es.net/science-dmz/
[SDN-add]	http://www.opennetsummit.org/archives/apr12/heller-mon-intro.pdf
[SDN-Arch]	SDN Architecture Overview: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf
[SDNCentral]	https://www.sdncentral.com/education/nfv-insiders-perspective-part-1-goals-history-promise/2013/09/
[SDN-NBI-Std]	Schneider, F.,“Challenges for the success of SDN and NFV (from a standardization perspective)”, http://www.ieice.org/~nv/nvs2013/nvs3-is1-schneider.pdf
[SDN-ONF]	https://www.opennetworking.org/sdn-resources/sdn-definition
[SDNv1]	OpenFlow Switch Specification 1.0.0, Open31 December 2009. Open Networking Foundation.
[SDNv2C]	http://www.plexxi.com/2014/11/maturity-evolution-sdn/#sthash.9JGaF1jl.dpbs
[Shenker]	https://www.sdncentral.com/news/scott-shenker-preaches-revised-sdn-sdnv2/2014/10/
[Sherry et al.]	Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., & Sekar, V. (2012). “Making middleboxes someone else’s problem”*Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication - SIGCOMM ’12* (p. 13). New York, New York, USA: ACM Press. doi:10.1145/2342356.2342359
[SPARQL]	SPARQL 1.1 Query Language. Technical report, W3C, 2013.
[Virtuoso]	Erling, O., and Mikhailov, I. RDF Support in the Virtuoso DBMS. In Auer, S., Bizer, C.Müller, and Zhdanova, A.V., editors, Conference on Social Semantic Web, volume 113 of LNI, pages 59–68. GI, 2007.
[WSGI]	P.J. Eby, Python Web Server Gateway Interface v1.0.1 – PEP-3333, https://www.python.org/dev/peps/pep-3333/
[YouTube-demos]	http://www.youtube.com/user/OpenNaaS

Glossary

AA	Authentication and Authorisation
AcMaR	Accounting Management Resource
aCSP	academic Cloud Service Provider
ACL	Access Control List
ADN	Application Defined Networking
AJAX	Asynchronous JavaScript and XML
AG	Aggregator
AKLP	Adapted k-longest path
AKSP	Adapted k-shortest path
ALP	Adapted Longest Path
API	Application Programming Interface
ARP	Address Resolution Protocol
ASU	Application Support Units
B2B	Business to Business
BCC	Building Cloud Competencies
BFD	Bidirectional Forwarding Detection
BoD	Bandwidth on Demand
BW	BandWidth
CaaS	Communications as a Service
capex	capital expenditure
CCM	Continuity Check Messages
CDN	Content Distribution Network
CE	Content Engine
CEF	Cisco Express Forwarding
CGI	Common Gateway Interface
CMaaS	Cloud Management as a Service
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRUD	Create Read Update and Delete
CSP	Cloud Service Provider
DaaS	Desktop as a Service
DC	Data Centre
DCN	Dynamic Circuit Network
DDoS	Distributed Denial of Service
E2E	Exchange to Exchange
EAA	Embedded Automation Architecture
EG	Expert Group
eTOM	enhanced Telecomunications Operations Map

EPL	Eclipse Public License
ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance, Security
FDB	Forwarding DataBase
FI-PPP	Future Internet Public Private Partnership
ForCES	Forwarding and Control Element Separation
FP	Feature Provider
FSPF	Fabric Shortest Path First
FSFW	Flowspace FireWall
FV	FlowVisor
FW	FireWall
GN3	GÉANT Network 3 project
GN3plus	GÉANT Network 3 plus project
GOFF	GÉANT OpenFlow Facility
gOCX	GÉANT Open Cloud Exchange
GRE	Generic Routing Encapsulation
GTS	Global Task Scheduling
GUI	Graphical User Interface
HD	High Definition
HDD	Hard Disk Drive
HNX	Helix Nebula
HPC	High Performance Computing
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	HardWare
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
ICN	Information Centric Networking
IDS	Inter-Domain Controller
IPS	Intrusion Prevention System
IPv4	Version 4 of the Internet Protocol (StB IETF)
IPv6	Version 6 of the Internet Protocol (StB IETF)
IXP	Internet eXchange Point
IPFIX	Internet Protocol Flow Information Export
IPMI	Intelligent Platform Management Interface
IRTF	Internet Research Task Force
ISG	Industry Specification Group
JRA2	Joint Research Activity 2
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KPI	Key Performance Indicators
L3VPN	Layer 3 Virtual Private Network
LACP	Link Aggregation Control Protocol
LINF	Linux Foundation
LLDP	Link Layer Discovery Protocol

MaaS	Management as a Service
MAC	Media Access Control
MANO	Management and Orchestration
MaR	Management Resource
MPLS	Multi-Protocol Label Switching
MSDP	Multicast Source Discovery Protocol
MTL	Message Transport Layer
MX	Mail eXchanger
NaaS	Network as a Service
NAT	Network Address Translation
NBI	Northbound Interface
NCS	Network Control System
NE	Network Element
NETCONF	Network Configuration Protocol
Net-HAL	Network Hardware Abstraction Layer
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NML	Network Markup Language
NMS	Network Management System
NOC	Network Operations Centre
NQA	Network Quality Analyser
NREN	National Research and Educational Network
NRM	Network Resource Manager
NSA	Network Service Agent
NSF	Network Service Framework
NSI	Network Service Interface
NSI-CS	Network Service Interface-Connection Service
OAM	Operations, Administration and Management
OCF	OFELIA Control Framework
OCX	Optical Cross-Connect
ODL	OpenDaylight
OESS	Open Exchange Software Suite
OF	Open Flow
OF2NF	OpenFlow to NetFlow
OGF	Open Grid Forum
ONF	Open Networking Foundation
ONOS	Open Networking Operating System
opex	Operational Expenditure
OPNFV	Open Platform for NFV
OSGi	Open Source Gateway Initiative
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OTT	Over The Top
OVS	Open vSwitch
OVX	OpenVirteX

P2P	Point to Point
PaaS	Platform as a Service
pCSP	public Cloud Service Provider
PeMaR	Performance Management Resource
PER	Performance and Portability
PoC	Proof of Concept
PoMaR	Provisioning Management Resource
PoP	Point of Presence
PR	Physical Resource
PrMaR	Provisioning Management Resource
QoE	Quality of Experience
QoS	Quality of Service
RAG	Red Amber Green
REL	Reliability and Availability
ROADM	Reconfigurable Optical Add-Drop Multiplexer
RC	Resource Consumer
REN	Research and Educational Network
REST	REpresentational State Transfer
RMON	Remote Network Monitoring
RO	Resource Operator
RP	Resource Provider
RPM	Resource Performance Management
RPC	Remote Procedure Call
SaaS	Software as a Service
SBI	Southbound Interface
SDA	Service Definition Agreement
SDK	Software Development Kit
SDN	Software Defined Networking
SDNRG	Software Defined Networking Research Group
SDNtrap	SDN Traffic Redirection Application
SDO	Standard Development Organization
SLA	Service Level Agreement
SME	Subject Matter Expert
SNMP	Simple Network Management Protocol
SP	Service Provider
SQM	Service Quality Management
STP	Service Termination Points
SWA	Software Architecture
T	Task
TCP/UDP	Transmission Control Protocol / User Datagram Protocol
TEBD	Time-Evolving Block Decimation
TTP	Trusted Third-Party Service
TrMaR	Trouble Management Resource
UDP	User Datagram Protocol
UI	User Interface

uRA	Ultimate Requester Agent
uPA	Ultimate Provider Agent
URI	Uniform Resource Identifier
URN	Uniform Resource Name
vCPE	virtual Customer Premise Equipment
VI	Virtual Infrastructure
VLAN	Virtual Local Area Network
VM	Virtual Machines
VNF	Virtualised Network Functions
VPS	Virtual Private Server
VR	Virtual Resource
WoR	Worker Resource
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language
XSD	XML Schema Definition